

Java学习群

72030155

进群可以免费获取视频教程以及每日免费听老师讲课

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间：可以联系群主获取更多大型企业内部技术教程。

前言

随着网络的日益普及和社会信息化程度的提高,越来越多的软件开发人员需要开发 Web 应用程序。Java EE Web 开发技术以其开放性、灵活性、安全性和技术成熟度,赢得了广大编程爱好者的青睐,并且在目前企业级 Web 应用系统开发领域占领了很大的市场,取得了许多成功的案例。

本书起源

尽管 Java EE 开发人员的队伍在不断壮大,但是 Java EE 开发目前不得不面临这样的困境:经过高校或者培训机构的教育学习,大部分开发人员都能够熟练掌握 Java EE 技术的基本语法,但是在解决实际项目开发中遇到具体问题时仍然感到无从下手,力不从心。另外,Java EE 技术是个充满活力的技术,新的技术规范和标准不断推出,刚刚熟悉了 EJB2 的你,不得不马上了解新推出的 EJB3;才搞懂 JSF 技术内幕,Hibernate3 又在向你招手。如何应对这些层出不穷的新技术带来的挑战和机遇,往往令 Java EE 初学者顾此失彼,心神疲惫。

造成上述现象的原因是由 Java EE Web 应用开发自身的特点决定的。企业系统的开发,本身就是一个复杂的工程,涉及需求分析、数据建模、系统架构、开发技术选型等多个方面,这些都要求开发人员必须在实践中不断学习体会,逐步提高。

Java 技术的开放性决定了 Java EE 编程技术的更新速度较其他开发技术要快得多,这也是 Java 技术自身的优势和魅力所在。所谓“万变不离其宗”,无论 Java Web 应用开发技术如何推陈出新,但 Web 应用的基本逻辑架构是不变的,即整个 Web 应用体系分为表现层、业务层和持久层,不同的 Java 技术分别服务于不同的 Web 应用结构层面,共同实现 Web 应用系统。把握住这个本质,开发人员便可以“以不变应万变”,从容面对新技术的挑战。

本书以 Java EE Web 应用系统的逻辑架构为主线,通过多个典型工程实例分别对 Java Web 应用各个层次上的开发技术和编程技巧进行讲解,并对同一实例的多个解决方案进行对比分析,切实帮助开发人员提高 Java EE Web 开发水平。

本书主要内容

本书各章节的内容安排如下。

第 1 章 构建动态 Web 页面:通过多个示例演示构建 Java Web 应用表现层的各种技术。

第 2 章 服务器与客户端的交互:演示客户端与服务器之间进行交互的基本方法和技术。

第 3 章 管理和维护应用程序的状态:通过多个示例演示 Java EE Web 开发中如何维护应用程序状态信息。

第 4 章 访问企业信息资源:演示 Java Web 应用如何与邮件服务器、关系数据库等外部信息系统交互。

第 5 章 开发 Web 高级功能特性:演示如何实现 Web 应用中常见的打印、图表、上传、下载、日志、国际化等高级功能特性。

第 6 章 应用架构:演示如何基于 Java 重量级架构及 Struts、Hibernate、Spring 等轻量级架构开发企业应用系统。

本书特色

与市场上的同类图书相比，本书具有以下特色：

1. 贴近实战，指导性强

书中所有示例都是作者结合多年教学实践和实际工程项目经验严格挑选的，力求简洁明了，切中要害，使读者能够快速理解并运用到实践中去。全部示例代码都经过严格测试，确保能够正确运行。

2. 思路清晰，结构严谨

书中内容遵循由浅入深、循序渐进的原则，分别按照 Web 应用特定层次（包括表现层、业务层和持久层）实现技术，高级开发特性及系统架构的顺序组织内容，便于开发人员学习理解。对于每个示例，按照学习目的、背景问题、解决方案、知识链接、实现步骤、运行结果、讨论分析的顺序进行讲解，使得初学者能够迅速掌握开发要领。

3. 多案并举，深入分析

避免一般实例图书中仅仅围绕实例实现堆砌代码的弊端，而是从整个 Java EE 编程技术的高处出发，对同一实例提出多个解决方案，并对不同解决方案各自的优缺点进行讨论分析，深入点评，使得开发人员能够尽快领会各种 Java EE 编程技术的实质和适用场景，从而切实提高解决实际问题的能力。

适用读者

本书适用于对 Java EE 编程有一定了解，但不知道如何开发实际 Web 应用系统的初学者，本书也适合希望掌握 Java EE 编程高级技巧，提高 Java EE 编程能力的中高级初学者。本书中的每个实例都按照知识点进行了索引，因此，本书也可以作为开发人员案头的参考手册，随时进行查阅。

使用与反馈

为方便广大读者使用本书学习 Java EE Web 编程，本书附赠的光盘中包含了所有例程的源代码及关于如何使用这些源代码的详细指导和说明。

由于作者水平有限，加之编写时间仓促，书中难免出现错误和不足。对于书中的任何问题，请发 E-mail 至邮箱：haoyulongsd@163.com。

致谢

在本书的编写过程中，得到众多老师的指导和帮助。感谢解放军理工大学的程宝义教授、张宏军教授、吴耀平高工，他们为本书提供了良好的技术支持。感谢周旋、尹鹏飞、周铭、张志杰、姜波，他们参与了本书的部分编辑修改工作，并对本书的内容组织提供了建设性的意见。感谢本书的编辑，北京交通大学出版社的谭文芳老师，没有她的辛勤劳动，本书不可能出版。感谢我的父母，在我写作的过程中给我无微不至的关怀。还要特别感谢我的女友季平，正是她的关心和鼓励，使我能够克服各种困难，确保本书能够顺利完成。

郝玉龙

2008 年 1 月

目 录

第1章 构建动态 Web 页面	1
例程 1-1: 利用 Servlet 显示动态日期	1
解决方案	1
讨论与思考	3
知识点索引	3
例程 1-2: 利用 JSP 表达式显示动态格式文本	3
解决方案	3
讨论与思考	5
知识点索引	5
例程 1-3: 利用 JSP 标准标记库和表达式语言显示动态新闻	5
解决方案	5
讨论与思考	8
知识点索引	9
例程 1-4: 利用自定义标记控制网页图片显示	9
解决方案	9
讨论与思考	15
知识点索引	16
例程 1-5: 利用 Applet 在 Web 页面实现动画时钟	17
解决方案	17
讨论与思考	28
知识点索引	28
例程 1-6: 利用 JSP 与 Flash 实现用户登录和注册模块	28
解决方案	29
讨论与思考	37
知识点索引	38
例程 1-7: 利用 JavaScript 脚本实现奥运倒计时日历	38
解决方案	38
讨论与思考	42
知识点索引	43
例程 1-8: 利用 XML、CSS 和 XSL 显示食谱信息	43
解决方案 1: 利用 CSS 显示食谱 XML 文件的内容	43
解决方案 2: 利用 XSL 显示食谱 XML 文件的内容	45
解决方案 3: 利用 xalan 动态绑定 XML 和 XSL	47
讨论与思考	48
知识点索引	49
例程 1-9: 利用标准标记库显示本地化信息	49

解决方案	49
讨论与思考	51
知识点索引	51
例程 1-10: 在 Web 页面中引入版权信息声明	51
解决方案 1: 使用 include 指令在 JSP 中包含版权信息	51
解决方案 2: 使用动作组件在 JSP 中包含版权信息	53
解决方案 3: 使用标准标记<c:import>在 JSP 中包含版权信息	54
解决方案 4: 在 Servlet 响应中包含版权信息	56
讨论与思考	57
知识点索引	58
例程 1-11: 实现带图形验证码的用户登录	58
解决方案	58
讨论与思考	62
知识点索引	62
例程 1-12: 利用 Ajax 实现网上智能订餐	62
解决方案	62
讨论与思考	71
知识点索引	73
本章小结	73
第 2 章 服务器与客户端的交互	74
例程 2-1: 奥运网上问卷调查	74
解决方案	74
讨论与思考	78
知识点索引	78
例程 2-2: 发送 PDF 文件到客户端浏览器	79
解决方案	79
讨论与思考	82
知识点索引	82
例程 2-3: 客户信息显示栏	82
解决方案	82
讨论与思考	83
知识点索引	84
例程 2-4: 获取服务器基本信息	84
解决方案	84
讨论与思考	86
知识点索引	86
例程 2-5: 横幅广告系统	86
解决方案	87
讨论与思考	97
知识点索引	98
例程 2-6: 利用过滤器限制客户端访问	98
解决方案	98

讨论与思考	103
知识点索引	103
例程 2-7: 多组件协作实现用户登录验证	103
解决方案	104
讨论与思考	108
知识点索引	108
本章小结	108
第 3 章 管理和维护应用程序状态	110
例程 3-1: 购物车	110
解决方案 1: 利用隐藏字段实现购物车	111
解决方案 2: 利用 URL 重写实现购物车	115
解决方案 3: 利用 Cookie 实现购物车	119
解决方案 4: 利用 Session 实现购物车	123
讨论与思考	127
知识点索引	128
例程 3-2: 聊天室	128
解决方案	128
讨论与思考	143
知识点索引	144
例程 3-3: 网站计数器	145
解决方案	145
讨论与思考	149
知识点索引	149
本章小结	150
第 4 章 访问企业信息资源	151
例程 4-1: 发送接收 E-mail	151
解决方案	151
讨论与分析	159
知识点索引	159
例程 4-2: 访问数据库	159
解决方案 1: 直接使用 JDBC 驱动访问数据库	159
解决方案 2: 利用 JDBC-ODBC 桥访问数据库	162
解决方案 3: 利用数据源和连接池技术访问数据库	165
讨论与思考	170
知识点索引	171
例程 4-3: 创建基于 XML 的网上论坛	171
解决方案	171
讨论与思考	191
知识点索引	192
例程 4-4: 访问体重检测 Web 服务	192
解决方案	192
讨论与思考	200

知识点索引	202
本章小结	202
第5章 开发 Web 高级功能特性	203
例程 5-1: 在 Web 页面显示统计图表	203
解决方案 1: 使用 Applet 显示统计图表	203
解决方案 2: 服务器端的图表解决方案	209
讨论与思考	214
知识点索引	214
例程 5-2: 为 Web 应用添加打印功能	215
解决方案 1: 利用 iText 组件打印 Web 表格	215
解决方案 2: 利用 JavaScript 脚本打印 Web 报表	218
讨论与思考	224
知识点索引	225
例程 5-3: 创建国际化的 Web 应用	225
解决方案 1: 为不同地区创建单独的页面资源	225
解决方案 2: 利用标准标记库自动绑定地区属性资源	229
讨论与思考	233
知识点索引	234
例程 5-4: 在 Web 应用中实现文件上传	234
解决方案 1: 利用流操作实现文件上传	234
解决方案 2: 利用 jspSmartUpload 组件实现上传	238
解决方案 3: 利用 common-upload 组件实现上传	241
讨论与思考	248
知识点索引	248
例程 5-5: 在 Web 应用中控制文件下载	248
解决方案 1: 利用文件流操作实现文件下载	249
解决方案 2: 利用 RequestDispatcher 实现文件下载	254
讨论与思考	255
知识点索引	256
例程 5-6: 为 Web 应用添加日志功能	256
解决方案 1: 利用服务器自身的日志功能	256
解决方案 2: 利用 log4j 实现日志功能	257
知识点链接	257
讨论与思考	260
知识点索引	261
本章小结	261
第6章 应用架构	262
预备知识: 软件架构基础	262
例程 6-1: 利用 EJB 实现公告发布系统	263
解决方案	263
讨论与思考	275
知识点索引	276

例程 6-2: 基于 Struts 构建新闻发布系统	276
解决方案	277
讨论与思考	289
知识点索引	289
例程 6-3: 基于 Struts、Spring 和 Hibernate 构建学生信息管理系统.....	290
解决方案	290
讨论与思考	303
知识点索引	303
本章小结	303
附录 A 开发环境的搭建	306
附录 B 知识点索引	311
参考文献	314

第 1 章 构建动态 Web 页面

条条大路通罗马

——谚语

动态 Web 页面是应用程序与用户进行动态交互的场所，因此构建动态 Web 页面是开发 Web 应用程序最基本的工作之一。本章将通过具体的示例演示 Java EE 体系中构建动态 Web 页面的各种基本方法和技巧，并对这些方法和技巧进行深入剖析和讨论，帮助 Java EE 开发人员彻底理清它们之间的脉络关系，从而为在实际的项目开发中具体采用何种技术构建 Web 页面奠定基础。

例程 1-1：利用 Servlet 显示动态日期

目的

- (1) 演示如何利用 Servlet 技术创建动态 Web 页面；
- (2) 日期函数的使用。

问题

如何将当前日期信息显示到 Web 页面，并且要求如果当前日期是周六、周日，则用红色字体显示日期信息。

解决方案

在 Servlet 负责处理客户端请求的方法(如 doGet()或 doPost())中，参数 HttpServletResponse 代表 Servlet 对客户端的响应。HttpServletResponse 的 PrintWriter 对象代表输出到客户端的显示信息。获取参数 HttpServletResponse 的 PrintWriter 对象，调用 PrintWriter 对象的 println()方法，将要显示的日期信息作为参数，即可将动态日期信息输出到客户端的 Web 页面。

示例代码

本例程的所有示例代码均在 netbeans 的工程 ServletOut 内。

程序 1-1: ShowDate.java

```
package com.example;

import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.Date;
```



```

public class ShowDate extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse
    response)
        throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        Date date=new Date();
        int day=date.getDay();
        String color="black";
        if(day==0||day==6) color="red";
        out.println("<html>");
        out.println("<head>");
        out.println("<title>显示动态日期</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<font color="+color+">");
        out.println("<h1>你好! 今天是" + date + "</h1>");
        out.println("</body>");
        out.println("</html>");
        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    public String getServletInfo() {
        return "Short description";
    }
}

```

程序说明：首先调用 `HttpServletResponse` 对象的 `setContentType("text/html; charset=UTF-8")` 方法设置 Servlet 对客户端响应的类型和编码格式，然后调用 `HttpServletResponse` 对象的 `getWriter()` 方法获得代表 Servlet 到客户端输出显示内容的 `PrintWriter` 对象，最后调用 `PrintWriter` 对象的 `println()` 方法将要显示的 HTML 页面一行行输出。

运行结果

说明：实例程序的开发运行环境为：Windows XP SP2 + JDK6+NetBeans 5.5+Sun Application Server PE 9+MySQL 5.0。其中 Sun Application Server PE 9 为 NetBeans 5.5 内置绑定的应用服务器。运行环境的安装配置过程可参见附录 A。

注意：如果不特别说明，本书中所有实例的运行环境均采用上述配置。

程序发布成功后，在浏览器地址栏输入“`http://localhost:8080/ServletOut/ShowDate`”，将得到如图 1-1 所示的运行结果。



图 1-1 利用 Servlet 显示动态日期

讨论与思考

Servlet 的工作模式是对客户端的请求进行处理, 并生成响应结果页面发送到客户端。利用 Servlet 构建动态页面的基本思路就是将 Servlet 对客户端请求的响应动态结果页面输出到客户端。其基本步骤如下: 获取代表 Servlet 到客户端输出显示内容的 `PrintWriter` 对象, 然后调用 `PrintWriter` 对象的 `println()` 方法将要显示的 HTML 页面逐行输出。

从例程 1-1 不难看出, 由于将 HTML 标记大量嵌入到 Java 代码中, 大大降低了程序的可读性, 同时要求程序开发人员必须熟练掌握 HTML 标记语言的使用。另外, 利用 `println()` 方法输出字符串的过程中, 对于一些特殊字符 (如 “\” 等) 必须采用转义字符标志进行标注。因此页面显示内容一旦复杂, 则给程序开发人员带来无尽的痛苦。因此, Servlet 主要完成应用程序后台的业务逻辑处理, 它仅仅用来输出比较简单的内容, 显示动态页面的任务一般由 JSP 来完成。

知识点索引

Servlet; 日期函数。

例程 1-2: 利用 JSP 表达式显示动态格式文本

目的

演示使用 JSP 表达式输出动态信息。

问题

如何在 Web 页面显示字体从大到小渐变的文本 “Java EE 编程技术实例精解”。

解决方案

通过 JSP 表达式, 动态输出定制的 HTML 标记, 从而在 Web 页面实现动态格式文本的显示。

知识链接

JSP 表达式是一种特殊的 JSP 脚本。JSP 脚本是 `<%与%>` 之间用 Java 语言编写的代码块。如果 JSP

脚本以“=”开头，如`<%=Java 代码%>`形式，称为 JSP 表达式，它将 Java 代码的计算结果转换为字符串形式输出到客户端，因此，可以利用 JSP 表达式来生成动态 HTML 页面。

注意：JSP 表达式中 % 与 = 之间不能有空格且表达式后面不需要分号。

实现步骤

- (1) 创建 JSP 页面 `expression.jsp`。
- (2) 在 JSP 页面中插入 JSP 脚本实现循环。
- (3) 在循环中插入 HTML 标记来输出文本“Java EE 开发实例精解”。
- (4) 利用 JSP 表达式动态生成的 HTML 标记。

示例代码

本例程的所有示例代码均在 netbeans 的工程 ShowDate 内。

程序 1-2: `expression.jsp`

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
<head>
<title>在 JSP 页面显示动态格式文本</title>
</head>
<body bgcolor="#FFFFFF">
<% for(int i=1;i<=5;i++){%>
<h<%=i%>>Java EE 开发实例精解</h<%=i%>><br>
<%}%>
</body>
</html>
```

运行结果

程序发布成功后，在浏览器地址栏输入“`http://localhost:8080/ShowDate/expression.jsp`”，将得到如图 1-2 所示的运行画面。

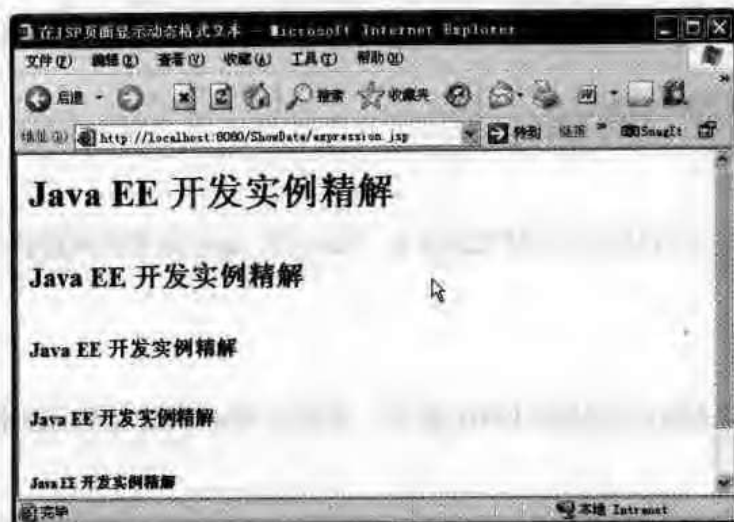


图 1-2 利用 JSP 表达式显示动态文本

讨论与思考

通过 JSP 脚本进行动态计算, 利用输出表达式将动态变量信息输出到页面生成动态标记是 JSP 页面编程的基本思路。JSP 脚本输出的动态内容可以是 HTML 页面中的任何内容, 包括 HTML 标记的名称、属性及标记体内容等信息。

由于 JSP 页面中可以直接嵌入 HTML 代码, 这样就避免了 Servlet 编程中大量的 `println()` 语句。但是使用 JSP 脚本生成动态内容仅适合于简单的动态 Web 页面实现, 它的缺点在于将 Java 代码与 HTML 混杂在一起, 如果页面的逻辑过于复杂, 则页面内容过于杂乱, 不利于提高页面开发效率, 也不便于开发过程中前端美工设计人员与后台业务逻辑人员的合理分工。最重要的, 这种开发模式将表现层的逻辑与业务逻辑紧密耦合在一起, 违背了 Java EE 多层体系架构的基本模式。

知识点索引

JSP 脚本; JSP 表达式。

例程 1-3: 利用 JSP 标准标记库和表达式语言显示动态新闻

目的

- (1) 演示 JSP 标准标记库的使用;
- (2) 演示表达式语言的使用;
- (3) JSP 与 JavaBean 的结合。

问题

网上新闻是网站常见的组件。每一条新闻由分类信息、标题和内容三部分组成。那么如何创建一个网上新闻组件实现在 Web 页面显示动态新闻信息?

解决方案

创建一个 JavaBean 来代表发布的新闻信息, 利用标准标记库的 `<c:out>` 将新闻内容输出到页面。

知识链接

1. JSP 标准标记库

JSP 标准标记库 (JSP Standard Tag Library, JSTL) 是一个实现 Web 应用程序中常见的通用功能的定制标记库集, 这些功能包括迭代和条件判断、数据管理格式化、XML 操作及数据库访问。通过为上述典型表现层任务提供标准实现, JSTL 使 JSP 开发人员可以专注于特定于应用程序的开发需求。

2. JSP 表达式语言

自 JSP2.0 以后, JSP 提供了一种新特性——JSP 表达式语言 (Expression Language, EL)。表达式语言允许开发人员使用简单的语法方便地从 JSP 页面访问数据, 而不需要使用 JSP 脚本或者 Java 表达式。

表达式语言定义了一些隐含对象以支持开发人员访问需要的应用程序数据。表达式语言定义的隐含

对象如表 1-1 所示。

表 1-1 表达式语言中的隐含对象

隐含对象	内 容
applicationScope	应用程序范围内的 scoped 变量组成的集合
cookie	所有 cookie 组成的集合
header	HTTP 请求头部, 字符串
headerValues	HTTP 请求头部, 字符串集合
initParam	全部应用程序参数名组成的集合
pageContext	当前页面的 javax.servlet.jsp. PageContext 对象
pageScope	页面范围内所有对象的集合
param	所有请求参数字符串组成的集合
paramValues	所有作为字符串集合的请求参数
requestScope	所有请求范围的对象的集合
sessionScope	所有会话范围的对象的集合

表达式语言支持“.”运算符来访问对象的属性信息, 同时还支持算术运算符、关系运算符和逻辑运算符, 以完成大多数的数据处理操作。此外, 它还提供了一个用于测试一个对象是否为空的特殊运算符。

实现步骤

(1) 将 JSTL 相关的类库 (jstl.jar 和 standard.jar) 添加到项目的类路径中。具体操作为: 在项目文件夹 jstlsample 下选中子文件夹【库】, 单击右键, 在弹出的快捷菜单中选择【添加库】选项, 弹出的对话框如图 1-3 所示。在对话框的列表中选择【JSTL 1.1】, 单击【添加库】按钮, 则成功地将 JSTL 相关的类库添加到项目的类路径中。

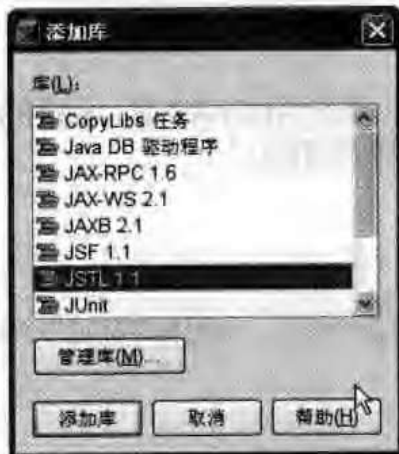


图 1-3 为 Web 工程添加 JSTL 库

- (2) 生成代表新闻内容的 JavaBean News。
- (3) 生成显示动态信息内容 JSP 页面 news.jsp。
- (4) 将下面的代码行添加到 news.jsp 的第一行以便在页面中使用标准标记库中的标记。

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
```

- (5) 在 news.jsp 中利用标准标记库和表达式语言显示动态新闻内容。

示例代码

本例程的所有示例代码均在 netbeans 的工程 jstlsample 内。

程序 1-3: News.java

```
package com.example;

import java.util.HashMap;
import java.util.Map;

public class News extends Object implements java.io.Serializable {

    private String topic;
    private HashMap values;

    public News() {
        topic="热烈欢迎进入 Java 开发人员阵营! ";
        values = new HashMap();
        values.put("netbeans", "netbeans6.0 正式发布!");
        values.put("Java", "Java 开发成员超过 3000 万!");
        values.put("北京", "奥运会准备一切就绪!");
    }

    public String getTopic(){
        return topic;
    }

    public Map getValues() {
        return this.values;
    }
}
```

程序说明：程序用来实现一个代表动态新闻的 **JavaBean**。其中动态新闻的内容采用一个 **HashMap** 对象来表示。

程序 1-4: news.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<jsp:useBean id="mynews" scope="page" class="com.example.News" />
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>通过表达式语言显示动态新闻</title>
</head>
<body>

<c:if test="${param.sayHello}">
<!-- Let's welcome the user ${param.name} -->
Hello ${param.name}!
</c:if>

<h2>${mynews.topic}</h2>
<font color="blue" >
<h3>
```



```

        开发工具类新闻: ${mynews.values.netbeans}<br>
        业界动态: ${mynews.values.Java}<br>
        奥运快讯: ${mynews.values["北京"]}<br>
        Eclipse: ${mynews.values.eclipse}
    </h3></font>
</body>
</html>

```

程序说明：首先使用`<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>`来引入标准标记库中的标记，然后调用 JSP 指令`<jsp:useBean>`将代表动态新闻的 JavaBean 引入页面。使用标准标记库中的`<c:if>`标记来控制页面内容的显示，利用表达式语言显示请求的参数信息及 JavaBean 的内容。其中利用表达式语言显示请求的参数信息时使用了隐含对象 `param`。而且，根据显示 JavaBean 中 `HashMap` 对象的值可以看出，“.”运算符可以嵌套使用。

运行结果

程序发布成功后，在浏览器地址栏输入“`http://localhost:8080/jstlsample/news.jsp?sayHello=true&name=haoyulong`”，将得到如图 1-4 所示的运行结果。



图 1-4 利用标准标记库和表达式语言显示动态新闻

讨论与思考

在例程 1-2 中使用 JSP 脚本实现了动态内容的显示。由于脚本元素依赖于嵌入页面中 Java 代码，所以对于使用脚本元素的 JSP 页面，其软件维护任务的复杂度大大增加了。另外，由于 JSP 在运行前首先编译成 Servlet，因此，由于脚本代码的错误造成的错误很难调试修改，大大影响 JSP 页面的开发效率。

而在本例程中，我们使用标准标记库和表达式语言来显示动态内容。通过将常用功能包装到定制标记库的标准集合中，JSTL 使 JSP 开发人员可以大大减少对脚本元素的需求，降低 JSP 页面相关的维护成本。

归纳起来，JSTL 具有以下优点。

- ✎ 在应用服务器之间提供了一致的接口，最大程度地提高了 Web 应用在各应用服务器之间的移植。
- ✎ 简化了 JSP 和 Web 应用程序的开发。
- ✎ 以一种统一的方式减少了 JSP 中的脚本代码数量，甚至可以实现没有任何脚本代码的程序。

尽管 JSTL 包含了大部分 Web 开发的通用功能,但开发人员也必须清醒地认识到,在某些情况下还是必须借助脚本元素来实现动态页面定制,因此,对于 JSP 脚本元素的编写技术还是要熟练掌握的。

知识点索引

JSTL: 表达式语言。

例程 1-4: 利用自定义标记控制网页图片显示

目的

- (1) 演示如何利用自定义标记定制页面显示效果;
- (2) Java 中对图像的操作。

问题

HTML 提供了标记实现在网页上图片的显示,但某些时候如对网站 Web 页面上图片的显示需要实现灵活控制,包括控制图片显示比例,决定是否加盖水印等,那么直接使用标记是不够的。那么如何灵活控制网站 Web 页面上图片的显示呢?

解决方案

开发一个自定义标记,利用自定义标记的属性和内容作为对图片的控制参数,实现对图片显示的灵活控制和管理。

知识链接

1. 什么是自定义标记

JSP 自定义标记可以看成是对 JSP 标准动作标记的一种扩展,正如 XML 是对 HTML 的一种扩展一样。通过自定义标记,将一些重复性的、与特定应用相关的业务逻辑以文档化标记的形式进行调用,维护 JSP 以文档为中心的开发方式,最大程度地降低 JSP 页面的代码含量,提高 JSP 页面可维护性。

自定义标记是用户定义的以 XML 形式表示的 JSP 语言元素。当一个包含自定义标记的 JSP 页面被转化为 Servlet 时,Web 容器自动调用自定义标记对应的标记处理程序来处理 JSP 页面。

2. 自定义标记表示

自定义标记的表示由下面几部分组成。

(1) 标记名称。标记名称必不可少,它由两部分组成:前缀,用来区分不同标记库,不同的标记库具有不同的前缀;后缀,用来区分统一标记库中的不同标记。前缀和后缀之间以冒号隔开。如自定义标记<mytag:hello>表示使用的是 mytag 标记库中的 hello 标记。

(2) 属性。就像 XML 文件中的标记一样,自定义标记可以有属性,可以通过属性来动态控制标记的行为。如标记<mytag:hello name="John">表示自定义标记 hello 带有一个 name 属性。

(3) 体内容:在标记的开始元素与结束元素之间的内容称为体内容。如自定义标记<mytag:hello name="John">welcome</mytag:hello>的体内容为 welcome。

实现步骤

(1) 创建辅助工具类 `ImgUtil`，实现对图片的基本操作，包括获取图片的显示尺寸，生成加盖水印的图片等。

(2) 创建标记处理程序 `ImageBodyTag`。它继承 `BodyTagSupport`，并实现对标记的属性和体内容的处理。

(3) 创建自定义标记库的描述文件 TLD（实际是一个 XML 文件）。具体操作为：在 Netbeans 中单击【新建】→【标记库描述符】，将得到【新建标记库描述符】对话框，如图 1-5 所示。在【TLD 名称】文本框中输入 TLD 文件的名称“`ImgTag`”，【URI】文本框中的内容为 TLD 文件的引用地址。【前缀】文本框中的内容为标记使用时的前缀，默认上述选项设置，单击【完成】按钮，TLD 文件创建完毕。



图 1-5 创建标记库描述文件

(4) 创建 JSP 页面对开发的自定义标记进行测试。

示例代码

本例程的所有示例代码均在 netbeans 的工程 `ImgTag` 内。

程序 1-5: `ImgUtil.java`

```
package com.example;

import com.sun.image.codec.jpeg.JPEGCodec;
import com.sun.image.codec.jpeg.JPEGEncodeParam;
import com.sun.image.codec.jpeg.JPEGImageEncoder;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.FileOutputStream;
```

```

import java.util.Date;
import javax.swing.ImageIcon;

public class ImgUtil {

    /** Creates a new instance of ImgUtil */
    public ImgUtil() {
    }
    //生成水印
    public String watermark(String filename){
        Color color=Color.RED;//前景色
        Color backgroundColor=Color.WHITE;//背景色
        String word="haoyulong";//水印文字
        int watermarkLocationX=100;
        int watermarkLocationY=100;
        int wordLocationX=50;
        int wordLocationY=50;

        ImageIcon srcFile=new ImageIcon(filename);//读入源文件
        Image srcImage =srcFile.getImage();//构造 Image 对象
        BufferedImage tagImage=new BufferedImage(srcImage.getWidth(null),srcImage.
        getHeight(null), BufferedImage.TYPE_INT_RGB);//构造图像缓存对象

        Graphics2D g=tagImage.createGraphics();
        g.setColor(color);//设置绘图笔的前景色
        g.setBackground(backgroundColor);//设置绘图笔的背景色
        g.drawImage(srcImage, 0, 0, null);//先绘制源图
        g.setFont(new Font("Arial",Font.BOLD,50));
        g.drawString(word,wordLocationX,wordLocationY); //添加文字
        g.dispose();

        Date now =new Date();
        int p=filename.indexOf(".");
        String t1=filename.substring(0,p);
        String t2=filename.substring(p);
        String tempname=t1+Long.toString(now.getTime())+t2;
        paint(tagImage,tempname);
        return tempname;
    }
    public void paint(BufferedImage tagImage,String tagFilename){
        try{
            FileOutputStream tagFile=new FileOutputStream(tagFilename);
            JPEGImageEncoder encoder=JPEGCodec.createJPEGEncoder(tagFile);
            JPEGEncodeParam param=encoder.getDefaultJPEGEncodeParam(tagImage);
            param.setQuality(50f, true);
            encoder.encode(tagImage, param);
            tagFile.close();
        }catch(Exception e){}
    }
    public int getWidth(String filename, String scale){

```



```

        ImageIcon srcFile=new ImageIcon(filename);//读入源文件
        Image srcImage =srcFile.getImage();//构造 Image 对象
        int width=srcImage.getWidth(null);
        float f=Float.parseFloat(scale);
        int temp=(int) (width*f);
        return temp;
    }

    public int getHeight(String filename, String scale){
        ImageIcon srcFile=new ImageIcon(filename);//读入源文件
        Image srcImage =srcFile.getImage();//构造 Image 对象
        int height=srcImage.getHeight(null);
        float f=Float.parseFloat(scale);
        int temp=(int) (height*f);
        return temp;
    }
}

```

程序说明：作为一个辅助类，主要实现对图像文件的操作，其中方法 `watermark(String filename)` 实现了给图片加水印的功能，方法 `paint(BufferedImage tagImage,String tagFilename)` 将内存中的图像文件输出到文件系统保存，方法 `getWidth(String filename, String scale)` 和方法 `getHeight(String filename, String scale)` 分别输出在特定缩放比例下图像显示时的宽度和高度。

程序 1-6: ImageBodyTag.java

```

package com.example;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.BodyTagSupport;

public class ImageBodyTag extends BodyTagSupport {
    //自定义标记属性
    private String heading = null;
    private String image =null;
    private String scale =null;
    private String watermark =null;
    public int doStartTag( ) throws JspException{
        try {
            int h = new Integer(heading).intValue( );
            if(! (h > 0 && h < 7))
                throw new JspException("The 'heading' attribute value must between 1
                    and 6 inclusive.");
        } catch (Exception e) { throw new JspException(e.getMessage( )); }
        return EVAL_BODY_BUFFERED;
    }

    public int doEndTag( ) throws JspException {
        JspWriter out = pageContext.getOut( );
        String realpath=pageContext.getServletContext().getRealPath("index.jsp");
        realpath=pageContext.getServletContext().getRealPath("/");
        // String imgDir = ((HttpServletRequest) pageContext.getRequest( )).
    }
}

```

```

getContextPath() + "/" ;
// String imgDir = realpath + "\\";
// 获取体内容
String message = getBodyContent().getString().trim();
try{
    //输出体内容
    String filename = realpath + image;
    ImgUtil iu = new ImgUtil();
    int width = iu.getWidth(filename, scale);
    int height = iu.getHeight(filename, scale);
    watermark = watermark.toUpperCase();
    if(watermark.equals("TRUE") | watermark.equals("YES")){
        filename = iu.watermark(filename);
    }
    out.println("<H" + heading + ">" +
        message.toUpperCase() + "</H" + heading + "><br>" + "<img src=\"" +
        filename + "\" width=\"" + width +
        "\" height=\"" + height + "\" align=\"left\">" );
} catch (java.io.IOException io) {}
return EVAL_PAGE;
} //doEndTag
//methods designed to set attribute values
public void setheading(String level){
    this.heading = level;
}
public void setImage(String name){
    this.image = name;
}
public void setScale(String scale){
    this.scale = scale;
}
public void setWatermark(String watermark){
    this.watermark = watermark;
}
//清空变量，避免标记重用产生问题
public void release() {
    heading = null;
    image = null;
    scale = null;
    watermark = null;
}
}

```

程序说明：作为标记处理器，程序实现了对自定义标记的翻译处理。程序继承了类 `BodyTag`，使得标记处理器可以处理标记的体内容。实现自定义标记处理的方法主要是两个方法：`doStartTag()` 实现对标记属性的校验功能；`doEndTag()` 方法实现对自定义标记显示内容的定制，它调用属性 `pageContext` 的 `getOut()` 方法获得代表输出内容的对象 `JspWriter`，进而实现对标记输出内容的控制。值得一提的是，利用属性 `pageContext` 还可以获得与包含自定义标记相关的 `HttpRequest`、`HttpResponse`、`HttpSession` 和 `ServletContext` 等对象。

程序 1-7: ImgTag.tld

```

<?xml version="1.0" encoding="UTF-8"?>
<taglib version="2.0" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xml/
ns/j2ee web-jsptaglibrary_2_0.xsd">
  <tlib-version>1.0</tlib-version>
  <short-name>imgtag</short-name>
  <uri>/WEB-INF/ImgTag</uri>
  <tag>
    <name>MyImg</name>
    <tag-class>com.example.ImageBodyTag</tag-class>
    <body-content>JSP</body-content>
    <description>This tag writes a Image inside the JSP.</description>
    <attribute>
      <name>heading</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
      <description>The heading level for the Image; 1 through 6.</description>
    </attribute>
    <attribute>
      <name>image</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
      <description>The name for the image.</description>
    </attribute>
    <attribute>
      <name>scale</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
      <description>The image diaplay scale </description>
    </attribute>
    <attribute>
      <name>watermark</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
      <description>是否显示水印 </description>
    </attribute>
  </tag>
</taglib>

```

程序说明：作为与其他 Java EE 组件的交互接口，主要用来对自定义标记库中的标记进行描述，包括自定义标记的名称，实现类，属性等信息。

程序 1-8: index.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ taglib uri="/WEB-INF/ImgTag.tld" prefix="MyImg" %>
<html>

```

```

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>JSP Page</title>
</head>
<body>

<MyImg:MyImg heading="2"scale="0.5" watermark="yes" image="rose.jpg" > 你是我的玫瑰花</MyImg:MyImg>
<MyImg:MyImg heading="4"scale="0.3" watermark="no" image="rose.jpg" > 你是我的玫瑰花</MyImg:MyImg>

</body>
</html>

```

程序说明：用来演示对自定义标记库的调用。共分两个步骤：第一步通过指令`<%@ taglib uri="/WEB-INF/ImgTag.tld" prefix="MyImg" %>`引入要使用的自定义标记库；第二步就可以直接使用开发的自定义标记了，如本例中的`<MyImg:MyImg heading="2"scale="0.5" watermark="yes" image="rose.jpg" > 你是我的玫瑰花</MyImg:MyImg>`。

运行准备

在运行示例程序之前，在 Web 应用程序的根目录下放置一张图片作为演示的素材（在本例中示例图像的文件名为 `rose.jpg`）。

运行结果

程序发布成功后，在浏览器地址栏输入“`http://localhost:8080/ImageTag/`”，将得到如图 1-6 所示的运行画面，左边一幅是加盖水印但显示比例较大的图像，右边一幅是没有加盖水印但显示比例较小的图像。



图 1-6 利用自定义标记控制图片显示

讨论与思考

JSP 自定义标记可以看成是对 JSP 标准动作标记的一种扩展，通过自定义标记，将一些重复性的、

与特定应用相关的业务逻辑以文档化标记的形式进行调用,减少了 JSP 页面的代码含量,提高 JSP 页面的可维护性。

自定义标记的开发者一般都是那些对 Java 编程语言非常精通,而且对数据访问和企业级服务访问都非常熟悉的程序员,这就使得页面设计者可以将精力放在界面设计上,直接使用自定义标记而不去关注复杂的商业逻辑成为可能。同时,自定义标记将那些重复工作进行封装,从而大大提高了生产力,而且可以使得标记可用于不同的项目中,完美地体现了软件复用的思想。

使用自定义标记库有以下优点。

(1) 易于安装在多个项目上

标记很容易从一个 JSP 项目迁移到其他项目。一旦建立了一个标记库,则只需要将相关的文件打包为一个 jar 文件,就可以在任何的 JSP 项目中重新使用这些 JSP 标记。

(2) 良好的扩展性

可以无限制地扩展和增加标记库的功能。

(3) 容易维护

标记库使得基于 JSP 的 Web 应用程序非常易于维护,原因如下。

- ✎ 标记应用简单,对任何人而言都很容易理解、使用。
- ✎ 所有的程序逻辑代码都集中放在标记处理器或 JavaBeans 中。这意味着在升级代码时,无须对每个使用该代码的 JSP 页面进行修改,只需要修改特定的代码文件便可。
- ✎ 如果需要加入新的功能,无须修改任何已经存在的页面,可以在标记中加入额外的属性,从而引进新的行为,而其他旧的属性不变,这样所有旧的页面还可以正常工作。例如,有一个让所有文本变红的标记:

```
<RedText>My Text</RedText>
```

但在后来项目中,又想让红色变暗。可以保留原有的标记,只要为其增加一个新的属性: shade, 如下所示:

```
<RedText shade="true">My Text</RedText>
```

所有旧的标记仍然可以产生红色的文本,但现在可以使用同一标记来产生变暗的红色文本了。

- ✎ 标记提升了代码的重用性:那些经过多次测试和使用的代码肯定具有更少的 Bug。所以,使用定制标记的 JSP 页面也同样具有更少的缺陷,维护起来自然方便多了。

(4) 快速的开发时间

标记库提供一个简单的方式来重用代码。通过重用代码,可以花费更少的时间来做开发,更多的时间可以用在设计 Web 应用上。标记库的接口也很简单,非常容易做扩展、使用和调试。

使用 JSP 标记库需要特别注意的一点是,对于标记库的开发要特别小心,代码必须经过严格测试。因为 JSP 标记库的重用性,使得引用它的外部环境变得非常复杂,开发中的疏忽将会造成莫名其妙的错误,而且由于标记库是封装发布的,使得程序调试变得更加困难。

知识点索引

自定义标记: Java 图像处理。

例程 1-5：利用 Applet 在 Web 页面实现动画时钟

目的

演示如何利用 Applet 实现高级用户界面。

问题

如何在 Web 页面通过一个动画时钟显示当前时间，并且用户可以与时钟进行交互，如调整时钟显示时间的时区设置等。

解决方案

利用 Java Applet 实现动画时钟，然后将 Applet 嵌入 Web 网页，通过嵌套在<applet>标记中的<param>标记将对 Applet 的控制信息传递到 Applet，也可以通过 JavaScript 脚本来直接调用 Applet 的方法。

实现步骤

- (1) 创建显示动画时钟的 Applet ClockApplet。
- (2) 创建引用 Applet 的静态页面 test.html。
- (3) 创建引用 Applet 的 JSP 页面 index.jsp。

示例代码

本例程的所有示例代码均在 netbeans 的工程 Appletclock 内。

程序 1-9: ClockApplet.java

```
import java.applet.Applet;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Event;
import java.awt.Font;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.MediaTracker;
import java.util.Date;
public class ClockApplet extends Applet
    implements Runnable
{

    static final Object colors[][];
    static final Object cityZone[][] = {
        {
            "夏威夷", "-10"
        }, {
            "阿拉斯加", "-9"
        }, {
```



```

        "墨西哥", "-6"
    }, {
        "格林威治", "0"
    }, {
        "巴黎", "1"
    }, {
        "莫斯科", "3"
    }, {
        "香港", "8"
    }, {
        "东京", "9"
    }, {
        "悉尼", "10"
    }
};

Thread threadClock;
boolean isAnalog;
Font fontText;
Color fontColor;
Color backColor;
Color hHandColor;
Color mHandColor;
Color sHandColor;
Color hPointColor;
Color mPointColor;
int xPoint[];
int yPoint[];
Image backImage;
MediaTracker tracker;
Image imageBuffer;
int fromGMT;
int currentZone;
int oldHour;
int oldMinute;
int oldSecond;
public void setCityZone(String s)
{
    currentZone = fromGMT;
    for(int i = 0; i < cityZone.length; i++)
        if(s.indexOf((String)cityZone[i][0]) != -1)
            currentZone = Integer.parseInt((String)cityZone[i][1]);
}
private Color findColor(String s, Color color)
{
    if(s != null)
    {
        s = s.toUpperCase();
        if(s.charAt(0) == '#')
            return new Color(Integer.parseInt(s.substring(1), 16));
        for(int i = 0; i < colors.length; i++)

```

```

        if(s.compareTo((String)colors[i][0]) == 0)
            return (Color)colors[i][1];
    }
    return color;
}

public void init()
{
    String s = getParameter("typeface");
    if(s == null)
        s = "Helvetica";
    int i;
    try
    {
        i = Integer.parseInt(getParameter("fontsize"), 10);
    }
    catch(NumberFormatException _ex)
    {
        i = 16;
    }
    fontText = new Font(s, 0, i);
    backColor = findColor(getParameter("backcolor"), backColor);
    fontColor = findColor(getParameter("fontcolor"), fontColor);
    hHandColor = findColor(getParameter("hhandcolor"), hHandColor);
    mHandColor = findColor(getParameter("mhandcolor"), mHandColor);
    sHandColor = findColor(getParameter("shandcolor"), sHandColor);
    hPointColor = findColor(getParameter("hpointcolor"), hPointColor);
    mPointColor = findColor(getParameter("mpointcolor"), mPointColor);
    tracker = new MediaTracker(this);
    s = getParameter("backimage");
    if(s != null)
    {
        backImage = getImage(getCodeBase(), s);
        tracker.addImage(backImage, 0);
    } else
    {
        backImage = null;
    }
    s = getParameter("analog");
    if(s != null && s.indexOf("false") > -1)
        isAnalog = false;
    else
        isAnalog = true;
    Date date = new Date();
    fromGMT = -Math.round((float)date.getTimezoneOffset() / 60F);
    currentZone = fromGMT;
}

public void start()
{

```

```
if(imageBuffer == null)
{
    Dimension dimension = size();
    imageBuffer = createImage(dimension.width, dimension.height);
}
if(threadClock == null)
{
    threadClock = new Thread(this);
    threadClock.start();
}

public void stop()
{
    if(threadClock != null)
    {
        threadClock.stop();
        threadClock = null;
        imageBuffer = null;
    }
}

public void run()
{
    try
    {
        tracker.waitForAll();
    }
    catch(InterruptedException _ex)
    {
        return;
    }
    while(Thread.currentThread() == threadClock)
    {
        repaint();
        try
        {
            Thread.sleep(50L);
        }
        catch(InterruptedException _ex)
        {
            return;
        }
    }
}

public void update(Graphics g)
{
    Date date = new Date();
    int i = (date.getHours() + currentZone) - fromGMT;
```



```

if(i >= 24)
    i -= 24;
if(i < 0)
    i += 24;
int j = date.getMinutes();
int k = date.getSeconds();
if(i != oldHour || j != oldMinute || k != oldSecond)
{
    Graphics g1 = imageBuffer.getGraphics();
    DrawBackground(g1);
    Dimension dimension = size();
    int l = dimension.width >> 1;
    int il = dimension.height >> 1;
    if(isAnalog)
    {
        double d = (double)Math.min(l, il) * 0.75D;
        double d1 = (double)Math.min(l, il) * 0.040000000000000001D;
        double d2 = 3.1415926535897931D * ((double)j / 30D + (double)k / 1800D);
        xPoint[0] = (int)Math.round((double)l - 2D * d1 * Math.sin(d2)) - 1;
        xPoint[1] = (int)Math.round((double)l - d1 * Math.cos(d2));
        xPoint[2] = (int)Math.round((double)l + d * Math.sin(d2)) + 1;
        xPoint[3] = (int)Math.round((double)l + d1 * Math.cos(d2));
        yPoint[0] = (int)Math.round((double)il + 2D * d1 * Math.cos(d2)) - 1;
        yPoint[1] = (int)Math.round((double)il - d1 * Math.sin(d2));
        yPoint[2] = (int)Math.round((double)il - d * Math.cos(d2)) + 1;
        yPoint[3] = (int)Math.round((double)il + d1 * Math.sin(d2));
        g1.setColor(mHandColor);
        g1.fillPolygon(xPoint, yPoint, 4);
        if(j < 30)
            g1.setColor(Color.white);
        else
            g1.setColor(Color.black);
        g1.drawLine(xPoint[0], yPoint[0], xPoint[1], yPoint[1]);
        g1.drawLine(xPoint[1], yPoint[1], xPoint[2], yPoint[2]);
        if(j < 30)
            g1.setColor(Color.black);
        else
            g1.setColor(Color.white);
        g1.drawLine(xPoint[2], yPoint[2], xPoint[3], yPoint[3]);
        g1.drawLine(xPoint[3], yPoint[3], xPoint[0], yPoint[0]);
        double d3 = 3.1415926535897931D * ((double)i / 6D + (double)j / 360D);
        d = (double)Math.min(l, il) * 0.5D;
        d1 = (double)Math.min(l, il) * 0.050000000000000003D;
        xPoint[0] = (int)Math.round((double)l - 2D * d1 * Math.sin(d3)) - 1;
        xPoint[1] = (int)Math.round((double)l - d1 * Math.cos(d3));
        xPoint[2] = (int)Math.round((double)l + d * Math.sin(d3)) + 1;
        xPoint[3] = (int)Math.round((double)l + d1 * Math.cos(d3));
        yPoint[0] = (int)Math.round((double)il + 2D * d1 * Math.cos(d3)) - 1;
        yPoint[1] = (int)Math.round((double)il - d1 * Math.sin(d3));
        yPoint[2] = (int)Math.round((double)il - d * Math.cos(d3)) + 1;
    }
}

```

```

yPoint[3] = (int)Math.round((double)i1 + d1 * Math.sin(d3));
gl.setColor(hHandColor);
gl.fillPolygon(xPoint, yPoint, 4);
if(i >= 0 && i <= 6 || i >= 12 && i <= 18)
    gl.setColor(Color.white);
else
    gl.setColor(Color.black);
gl.drawLine(xPoint[0], yPoint[0], xPoint[1], yPoint[1]);
gl.drawLine(xPoint[1], yPoint[1], xPoint[2], yPoint[2]);
if(i >= 0 && i <= 6 || i >= 12 && i <= 18)
    gl.setColor(Color.black);
else
    gl.setColor(Color.white);
gl.drawLine(xPoint[2], yPoint[2], xPoint[3], yPoint[3]);
gl.drawLine(xPoint[3], yPoint[3], xPoint[0], yPoint[0]);
d = (double)Math.min(1, i1) * 0.75D;
gl.setColor(sHandColor);
double d4 = (3.1415926535897931D * (double)k) / 30D;
gl.drawLine(1, i1, (int)Math.round((double)1 + d * Math.sin(d4)),
    (int)Math.round((double)i1 - d * Math.cos(d4)));
} else
{
    gl.setFont(fontText);
    gl.setColor(fontColor);
    String s = (i >= 10 ? Integer.toString(i) : "0" + i) + ":" + (j >= 10 ?
        Integer.toString(j) : "0" + j) + ":" + (k >= 10 ? Integer.toString(k) :
        "0" + k);
    FontMetrics fontmetrics = gl.getFontMetrics();
    int j1 = dimension.height >> 1;
    if(j1 < 0)
        j1 = 0;
    int k1 = dimension.width - fontmetrics.stringWidth(s) >> 1;
    if(k1 < 0)
        k1 = 0;
    gl.drawString(s, k1, j1);
}
oldHour = i;
oldMinute = j;
oldSecond = k;
}
g.drawImage(imageBuffer, 0, 0, null);
}

public void paint(Graphics g)
{
    Dimension dimension = size();
    if(tracker.isErrorAny())
    {
        g.setColor(Color.white);
        g.fillRect(0, 0, dimension.width, dimension.height);
    }
}

```



```

        return;
    }
    if(tracker.checkAll(true))
        DrawBackground(g);
}

public boolean mouseDown(Event event, int i, int j)
{
    isAnalog = !isAnalog;
    return true;
}

private void DrawBackground(Graphics g)
{
    Dimension dimension = size();
    g.setColor(backColor);
    g.fillRect(0, 0, dimension.width, dimension.height);
    if(backImage != null)
    {
        int i = backImage.getWidth(this);
        int k = backImage.getHeight(this);
        if(i < 0 || k < 0)
            return;
        g.drawImage(backImage, dimension.width - i >> 1, dimension.height - k >>
            1, null);
    }
    if(isAnalog)
    {
        int j = dimension.width >> 1;
        int l = dimension.height >> 1;
        double d = (double)Math.min(j, l) * 0.90000000000000002D;
        for(int i1 = 1; i1 <= 12; i1++)
        {
            double d1 = 3.1415926535897931D * (0.5D - (double)i1 / 6D);
            int k1 = (int)Math.floor((double)j + d * Math.cos(d1));
            int l1 = (int)Math.floor((double)l - d * Math.sin(d1));
            g.setColor(hPointColor);
            g.fill3DRect(k1 - 2, l1 - 2, 4, 4, true);
        }
        for(int j1 = 1; j1 <= 60; j1++)
            if(j1 % 5 != 0)
            {
                double d2 = (3.1415926535897931D * (double)j1) / 30D;
                int i2 = (int)Math.floor((double)j + d * Math.cos(d2));
                int j2 = (int)Math.floor((double)l - d * Math.sin(d2));
                g.setColor(mPointColor);
                g.fill3DRect(i2 - 2, j2 - 2, 3, 3, false);
            }
    }
}

```

```

public ClockApplet()
{
    isAnalog = true;
    fontColor = Color.black;
    backColor = Color.lightGray;
    hHandColor = Color.blue;
    mHandColor = Color.blue;
    sHandColor = Color.black;
    hPointColor = Color.red;
    mPointColor = Color.lightGray;
    xPoint = new int[4];
    yPoint = new int[4];
    oldHour = -1;
    oldMinute = -1;
    oldSecond = -1;
}
static
{
    colors = (new Object[][] {
        new Object[] {
            "BLACK", Color.black
        }, new Object[] {
            "BLUE", Color.blue
        }, new Object[] {
            "CYAN", Color.cyan
        }, new Object[] {
            "DARKGRAY", Color.darkGray
        }, new Object[] {
            "GRAY", Color.gray
        }, new Object[] {
            "GREEN", Color.green
        }, new Object[] {
            "LIGHTGRAY", Color.lightGray
        }, new Object[] {
            "MAGENTA", Color.magenta
        }, new Object[] {
            "ORANGE", Color.orange
        }, new Object[] {
            "PINK", Color.pink
        }, new Object[] {
            "RED", Color.red
        }, new Object[] {
            "WHITE", Color.white
        }, new Object[] {
            "YELLOW", Color.yellow
        }
    });
}
}

```

程序说明：实现显示时间的时钟动画。Applet 实现了 Runnable 接口以利用多线程技术，以满足在时钟动画显示的同时还能够响应用户的请求。

程序 1-10: test.html

```
<html>
<head>
<title>Java Clock</title>
<SCRIPT LANGUAGE="JavaScript">

function timezone (indexs)
{
    document.javaClock.setCityZone (indexs);
}
</SCRIPT>
</head>

<body>
<h1 align=center><b>Java 时钟</b></h1>
<hr>
<center>
<FORM>
<SELECT name="lbSelect1">
<OPTION>当地时间
<OPTION>夏威夷
<OPTION>阿拉斯加
<OPTION>墨西哥
<OPTION>格林威治
<OPTION>巴黎
<OPTION>莫斯科
<OPTION>雅加达
<OPTION>香港
<OPTION>东京
<OPTION>悉尼
</SELECT>
<INPUT type="button" value="刷新"
        onClick="timezone
(document.forms[0].lbSelect1.options[document.forms[0].
lbSelect1.selectedIndex].text)">
</FORM>
<APPLET name="javaClock" code="ClockApplet.class" width=350 height=250>
<PARAM name="fontsize" value="36">
<PARAM name="backcolor" value="lightgray">
<PARAM name="fontcolor" value="black">
<PARAM name="hHandColor" value="blue">
<PARAM name="mHandColor" value="blue">
<PARAM name="sHandColor" value="green">
<PARAM name="hPointColor" value="red">
<PARAM name="mPointColor" value="lightgray">
<PARAM name="analog" value="true">
</APPLET>
```



```

</center>
</body>
</html>

```

程序说明：利用 HTML 文件调用 Applet 显示时钟动画。调用 Applet 的方式很简单，只需要使用标记<applet>即可。在标记<applet>内，可以进一步嵌套标记<param> 来向 Applet 传递参数初始化信息。另外，在 JavaScript 中，还可以直接调用 Applet 的方法。例如，在本例中。在 JavaScript 的方法 timrzone() 中，就直接调用了 Applet 的方法 setCityZone (indeks)来设置时钟的时区。

程序 1-11: index.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>

<head>
  <title>Java Clock</title>
  <SCRIPT LANGUAGE="JavaScript">

function timezone (indeks)
{
  document.javaClock.setCityZone (indeks);
}

  </SCRIPT>
</head>

<body>
<h1 align=center><b>Java 时钟</b></h1>
<hr>
<center>
  <FORM>
    <SELECT name="lbSelect1">
      <OPTION>当地时间
      <OPTION>夏威夷
      <OPTION>阿拉斯加
      <OPTION>墨西哥
      <OPTION>格林威治
      <OPTION>巴黎
      <OPTION>莫斯科
      <OPTION>雅加达
      <OPTION>香港
      <OPTION>东京
      <OPTION>悉尼
    </SELECT>
    <INPUT type="button" value="刷新"
      onClick="timezone
(document.forms[0].lbSelect1.options[document.
  forms[0].lbSelect1.selectedIndex].text)">
  </FORM>
  <jsp:plugin name="javaClock" type="applet" code="ClockApplet.class" >
    <jsp:params>
      <jsp:param name="fontsize" value="36" ></jsp:param>
      <jsp:param name="backcolor" value="lightgray" ></jsp:param>
      <jsp:param name="fontcolor" value="black" ></jsp:param>

```

```

<jsp:param name="hHandColor" value="blue" ></jsp:param>
<jsp:param name="mHandColor" value="blue" ></jsp:param>
<jsp:param name="sHandColor" value="green" ></jsp:param>
<jsp:param name="hPointColor" value="red" ></jsp:param>
<jsp:param name="mPointColor" value="lightgray" ></jsp:param>
<jsp:param name="analog" value="false" ></jsp:param>

</jsp:params>

<jsp:fallback>
    Plugin tag OBJECT or EMBED not supported by browser.
</jsp:fallback>
</jsp:plugin>

<PARAM name="" value="">
<PARAM name="backcolor" value="lightgray">
<PARAM name="fontcolor" value="black">
<PARAM name="hHandColor" value="blue">
<PARAM name="mHandColor" value="blue">
<PARAM name="sHandColor" value="green">
<PARAM name="hPointColor" value="red">
<PARAM name="mPointColor" value="lightgray">
<PARAM name="analog" value="true">
</APPLET>
</center>
</body>
</html>

```

程序说明：演示如何在 JSP 页面引用 Applet。在 JSP 页面中，使用指令<jsp:plugin>来引入 Applet，并可嵌套指令<jsp:params>和<jsp:param>给 Applet 传递初始化信息。

运行结果

程序发布成功后，在浏览器地址栏输入“http://localhost:8080/AppletClock/index.jsp”，将得到如图 1-7 所示的运行结果。



图 1-7 在 JSP 页面通过 Applet 显示动画时钟

在浏览器地址栏输入“http://localhost:8080/AppletClock/test.html”，将得到如图 1-8 所示的运行结果。



图 1-8 在 HTML 页面通过 Applet 显示动画时钟

讨论与思考

Applet 为开发复杂的 Web 应用用户界面提供了一个新的途径。使用 Applets 的优点有以下几个方面。

(1) Applet 可以提供比 Servlets 和 JSP 更加复杂的界面。相对于 HTML 组件，Applet 的组件更为丰富，特别是 Applet 的事件处理能力，更是提高了界面的智能化，大大丰富了用户体验。

(2) 因为 Applet 是从服务器上下载并运行在客户机上，Web 服务器不必支持 Java。这一点很重要，尤其是当开发人员为一个站点写一个 Web 应用，但又无权控制 Web 服务器时。

(3) Applet 可以进行本地校验，而不必在远端的服务器上进行校验。尽管也可以采用 JavaScript 加上 Servlet 或 JSP 来完成此功能。

(4) 在 Applet 下载完成后，浏览器到服务器的连接请求数目将会下降。因为大量的处理将在客户浏览器上完成。有助于减轻网络负载。有些信息还可以临时保存在客户端，当连线时再与服务器进行交互，这一点特别适合网络环境恶劣的情况。

但是使用 Applet 也具有明显的缺点，主要表现在以下几个方面。

(1) 相对于 Servlet 和 JSP，Applet 分发与测试是困难的。

(2) 依赖于客户机的浏览器是否允许 Java 运行。不同的浏览器版本支持不同版本的 JDK，并且一般不是最新的 JDK 版本。

(3) Applet 第一次启动很慢。因为客户机必须将它从服务器上下载。

知识点索引

JSP; Applet。

例程 1-6：利用 JSP 与 Flash 实现用户登录和注册模块

目的

演示 JSP 与 Flash 交互的技术。

问题

如何通过 JSP 和 Flash 来实现一个用户注册与登录的 Web 应用系统?

解决方案

利用 Flash 技术的 ActionScript 脚本的 LoadVars 类来实现与后端服务器组件之间的交互。

知识链接

ActionScript 是 Flash 文件内置的一种脚本编程语言,用于控制 Flash 页面与用户的动态的复杂交互,以及 Flash 与后端服务器组件之间的交互。目前 ActionScript 的最新版本为 3.0。关于 ActionScript 的详细信息可以查阅相关资料。

ActionScript 的 LoadVars 类用于在 Flash 应用程序和服务器之间传输变量。LoadVars 类使开发人员可以将对象中的所有变量发送到指定的 URL 中,并且可以将指定 URL 中的所有变量加载到某个对象中。

LoadVars 类使用 load()、send()和 sendAndLoad()方法与服务器进行通信。在与服务器进行通信时,LoadVars 传输 ActionScript 变量的名称和值对。

实现步骤

- (1) 利用 Flash MX 2004 或其他 Flash 开发工具开发 Flash 文件(源文件为 login fla)。
- (2) 在 Flash 中对应的帧添加相应的 ActionScript 脚本。
- (3) 将完成脚本调试的 Flash 输出为 login.swf 并添加到项目文件夹下。
- (4) 实现数据库操作的组件 DBConnection。
- (5) 实现登录验证的 JSP 页面 login.jsp。
- (6) 实现用户注册的 JSP 页面 add.jsp。

示例代码

本例程的所有示例代码均在 netbeans 的工程 flash 内。

程序 1-12: ActionScript 脚本 1 (位于登录帧下)

```
//实现登录验证的 ActionScript 脚本
stop();
//将 windows 组件设置为不可用,因为是做背景
win.enabled = false;
//新建 LoadVars 对象 loginData, 用来发送和接收数据
var loginData:LoadVars = new LoadVars();
//注册按钮
register.onRelease = function() {
    win.title = "JSP 与 Flash 交互示例";
    msg.text="";
    gotoAndStop(2);
};
//登录按钮
login.onRelease = function() {
    //判断用户名和密码是不是为空
```



```

if ((username.text == "") || (password.text == "")) {
    msg.text = "请正确输入用户名或密码!";
    //判断用户名和密码是否小于 8 位
} else if ((username.length<8) || (password.length<8)) {
    msg.text = "用户名和密码不能小于 8 位!";
} else {
    msg.text = "验证中...";
    //将用户名文本框的值赋给 loginData 对象的 username 变量
    loginData.username = username.text;
    //将用户密码文本框的值赋给 loginData 对象的 userpassword 变量
    loginData.userpassword = password.text;
    //使用 get 方法发送用户名和密码到 login.jsp 中验证;再返回给 loginData 对象
    loginData.sendAndLoad("login.jsp", loginData, "get");
}
};
//调用 LoadVars 对象的 onLoad 事件
loginData.onLoad = function(success) {
    //判断加载 login.jsp 是否成功
    if (success) {
        //这个是 JSP 中查询数据库中返回的值
        if (loginData.success == "true") {
            msg.text = "登录成功";
            username.text = "";
            password.text = "";
        } else {
            msg.text = "用户名或密码不正确";
            username.text = "";
            password.text = "";
        }
    } else {
        msg.text = "连接网络失败";
    }
};

```

程序说明：脚本主要与后台组件 login.jsp 交互，实现用户登录的校验功能。

脚本中首先创建一个 LoadVars 对象 loginData，用来发送和接收数据。在登录按钮的响应方法中，首先对输入的用户名和密码进行初步校验，然后将其加入到 loginData 对象，并调用 loginData 对象的 sendAndLoad() 将对象发送到后台服务器端的组件，其中，方法的第一个参数为服务器组件的地址，第二个参数为返回数据的存放变量，第三个参数为请求的处理方式。

为确保与后台服务器组件之间的时序，使用 LoadVars.onLoad 处理函数来确保脚本应用程序在加载数据之时（而不是之前）运行。关于登录的校验结果放置在对象变量 loginData 的 success 属性中，因此，可以直接采用点运算符进行访问。根据 success 属性的值，相应的在 Flash 页面显示关于登录的提示信息。

程序 1-13: login.jsp

```

//调用数据库访问组件，实现登录验证功能
<%@ page contentType="text/html; charset=utf-8" language="java" import="flash.*"
errorPage="" %>
<%
//获取 Flash 传过来的用户名和密码

```

```
String Username=request.getParameter("username");
String Password=request.getParameter("userpassword");
DBConnection db=new flash.DBConnection();
//将结果发送到 Flash 中
out.println("&success=" +db.login(Username,Password)+"&");
System.out.println("&success=" +db.login(Username,Password)+"&");

%>
```

程序说明：JSP 获取 ActionScript 脚本传递来的参数与获取普通网页传递来的数据的方法完全一致，也是调用请求的 `getParameter()`。随后创建一个数据库对象实例，根据传递的参数进行处理。由于 JSP 是被 ActionScript 脚本加载的，只需要利用 `out` 对象将要传递给 ActionScript 脚本信息输出即可。唯一需要特别注意的是：在参数名和值对的两端加上分割符号“&”。

程序 1-14：实现用户注册的 Actionscript 脚本（位于 flash 的注册帧下）

```
//用来实现用户注册脚本
stop();
//返回按钮
fanghui.onRelease = function() {
    win.title = "JSP 与 Flash 交互示例";
    password2.enabled=false;
    psconfirm.enabled=false;
    msg.text="";
    gotoAndStop(1);
};
//提交按钮
tijiao.onRelease = function() {
    //判断用户名和密码是不是为空
    if ((username.text == "") || (password.text == "")) {
        msg.text = "用户名和密码不能为空!";
        //判断用户名和密码是否小于 8 位
    } else if ((username.length<8) || (password.length<8)) {
        msg.text = "用户名和密码不能小于 8 位!";
    } else if (password.text!=password2.text) {
        msg.text = "两次输入的密码不一致!";
        username.text="";
        password.text="";
        password2.text="";
    }

    else {
        msg.text = "注册中...";
        //将用户名文本框的值赋给 loginData 对象的 username 变量
        loginData.username = username.text;
        //将用户密码文本框的值赋给 loginData 对象的 userpassword 变量
        loginData.userpassword = password.text;
        //使用 get 方法发送用户名和密码到 add.jsp 中查询;再返回给 loginData 对象
        loginData.sendAndLoad("add.jsp", loginData, "get");
    }
};
```



```

//调用 LoadVars 对象的 onLoad 事件
loginData.onLoad = function(success) {
    //判断加载 login.jsp 是否成功
    if (success) {
        //这个是 JSP 中查询数据库中返回的值
        if (loginData.reg > 0) {
            msg.text = "注册成功";
            username.text = "";
            password.text = "";
        } else {
            msg.text = "用户名已存在";
            username.text = "";
            password.text = "";
        }
    } else {
        msg.text = "连接网络失败";
    }
};

```

程序说明：脚本主要用来实现与后台组件 add.jsp 的交互，完成用户注册功能。

在注册按钮的响应方法中，首先对输入的用户名和密码进行初步校验，然后将其加入到 loginData 对象，并调用 loginData 对象的 sendAndLoad() 方法将对象发送到后台服务器端的组件，其中，方法的第一个参数为服务器组件的地址，第二个参数为返回数据的存放变量，第三个参数为请求的处理方式。

为确保与后台服务器组件之间的时序，使用 LoadVars.onLoad 处理函数来确保脚本应用程序在加载数据之时（而不是之前）运行。关于注册的结果信息放置在对象变量 loginData 的 reg 属性中。根据 reg 属性的值，相应的在 Flash 页面显示关于登录的提示信息。

程序 1-15: add.jsp

```

//用来实现注册功能
<%@ page contentType="text/html; charset=utf-8" language="java" import="flash.*"
errorPage="" %>

<%
    //获取 Flash 传过来的用户名和密码
    String Username=request.getParameter("username");
    String Password=request.getParameter("userpassword");
    DBConnection db=new DBConnection();
    out.println( db.register(Username>Password));

%>

```

程序说明：程序的基本设计思路与程序 1-13 完全一致，不再赘述。

程序 1-16: DBConnection.java

```

package flash;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

```

```

/**
 *
 * @author hyl
 */
public class DBConnection {

    /** 创建连接实例 */
    public DBConnection() {
    }

    public Connection getConnection() { //获取数据库链接
        java.sql.Connection conn;
        try{
            Class.forName("org.gjt.mm.mysql.Driver").newInstance();
            conn= java.sql.DriverManager.getConnection("jdbc:mysql://localhost/flash","root","root");

        } catch (java.sql.SQLException e){
            System.err.println(e.toString());
            return null;
        } catch (Exception ea){
            System.err.println(ea.toString());
            return null;
        }
        return conn;
    }

    public boolean login(String username,String password ){
        Connection conn=getConnection();
        try{
            PreparedStatement login_ps = conn.prepareStatement("SELECT * FROM user WHERE username = '" + username + "' AND password= '" + password + "'");
            //executeQuery()执行查询，将查询结果赋给新建的 login_rs 对象
            ResultSet login_rs = login_ps.executeQuery();
            //使用 login_rs.next() 判断是否有记录：有就返回 True，否则返回 False
            boolean login_success = login_rs.next();

            login_rs.close();
            login_ps.close();
            conn.close();
            return login_success;
        } catch (SQLException e){
            System.out.println(e.toString());
            return false;
        }
    }

    public String register(String username,String password ){
        Connection conn=getConnection();
        String result="";
    }
}

```



```

//新建一个查询表的对象
try{
    PreparedStatement chaxun_ps = conn.prepareStatement("SELECT * FROM user WHERE
    username = '" + username + "'");
    //executeQuery() 执行查询, 将查询结果赋给新建的 login_rs 对象
    ResultSet chaxun_rs = chaxun_ps.executeQuery();
    //使用 login_rs.next() 判断是否有记录; 有就返回 True, 否则返回 False
    boolean chaxun_success = chaxun_rs.next();
    //如果有记录就发送 reg 到 Flash 中, 告诉 Flash 用户名已存在
    if(chaxun_success){
        result("&reg=cunzai&");
    }else{
        PreparedStatement register_ps = conn.prepareStatement("INSERT INTO user
        (username,password) VALUES('" + username + "','" + password + "')");
        //如果插入成功 register_ps.executeUpdate() 将返回 1, 否则返回 0. 再将返回值发送到
        Flash 中.
        result("&reg="+register_ps.executeUpdate()+"&");
    }
    //关闭 chaxun_rs 对象
    chaxun_rs.close();
    //关闭 chaxun_ps 对象
    chaxun_ps.close();
    //关闭 conn 对象
    conn.close();
} catch(SQLException e){
    System.out.println(e.toString());
    result="connec database failed....";
}
return result;
}
}

```

程序说明: 程序主要实现与数据库的交互。共有三个方法, `getConnection()` 用来获取对数据库的连接, `login()` 用来实现登录校验功能, `register()` 用来实现注册功能。

程序 1-17: index.jsp

```

//用来包含 flash 动画。
<%@ page contentType="text/html; charset=utf-8"%>
<html>
<head><title> JSP 与 Flash 交互示例</title></head>
<body>
<OBJECT CLASSID=
    "clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" CODEBASE=
    "http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#
    version=6,040,0" width="380" height="225"
>
<PARAM name="movie" VALUE="login.swf">
<PARAM name="quality" VALUE="high">
<PARAM name="bgcolor" VALUE="#FFFFFF">

```

```

</OBJECT>
</body>
</html>

```

程序说明：页面中利用<object>对象来控制 Flash 动画 swf 文件的显示，其中，CLASSID 为 Flash 播放器的注册标志。

说明：完全可以直接访问 Flash 动画的 swf 文件，而不需要将其嵌在网页中访问。

运行准备

步骤 1：建立后台数据库

打开 MySQL 数据库，首先建立一个用户注册和登录系统专用的数据库 flash。系统中共有 1 个表格：用户信息表 user，用来存储用户的登录信息。表格的结构信息如表 1-2 所示。

表 1-2 用户登录表 user 的结构信息

字段名	字段类型	字段说明	备注
Id	Integer	用户 ID	主键，系统自增字段
username	Varchar (30)	用户登录名称	
password	Varchar (20)	用户密码	

提示：可以根据以上信息在数据库中建立相应表格。也可通过脚本文件 flash.sql 来还原数据库。MySQL 有很多客户端软件，如 EngInSite MySQL Client，可以帮助开发人员还原数据库。

步骤 2：添加数据库驱动包到工程的编译路径

要成功地访问后台数据库，必须将数据库驱动程序添加到工程的 classpath 路径下。具体操作步骤是：在【项目】视图下，选中项目文件夹【flash】下的【库】文件夹，单击右键，在弹出的快捷菜单中选中【添加 JAR/文件夹】选项，如图 1-9 所示。



图 1-9 为工程添加 Jar 文件

弹出【添加 Jar/文件夹】对话框，如图 1-10 所示。选择 MySQL 的 JDBC 驱动程序，单击【打开】按钮，则将数据库的驱动程序包添加到了工程的编译路径中。打开库文件夹，在下面可以看到新增的 MySQL 的 JDBC 驱动程序。



图 1-10 添加数据库驱动包

运行结果

应用部署成功后，在浏览器的地址栏输入“<http://localhost:8080/flash/>”，将得到如图 1-11 所示的运行结果。



图 1-11 用户登录界面

单击【注册】按钮，转到注册页面，如图 1-12 所示。



图 1-12 用户注册界面

输入注册信息后,单击【提交】按钮,显示出“注册成功”的提示信息。如图 1-13 所示,再返回到登录页面,按照注册的用户名和密码进行注册,将会实现成功登录,如图 1-14 所示。



图 1-13 用户注册成功



图 1-14 用户登录成功

讨论与思考

在网站的设计实现中,网站的前端页面绝大部分都是采用 HTML 文件的形式来实现的,即使是动态网站的建设,不管它的服务器端的脚本技术是采用 JSP、PHP 还是 ASP,它的基本原理也都是通过上述服务器端脚本技术在客户端输出动态生成的 HTML 标记。但使用 HTML 文件并不是建设网站的唯一途径,其他技术方式还有:如 XML 和 XSL 相结合的技术,示例网站如 <http://community.csdn.net/>; Flash 技术方式,如 <http://www.internetwork.com>。本例程重点演示如何使用 JSP 和 Flash 实现动态网站。

在上面的解决方案中,采用了 LoadVars 方式实现了 JSP 与 Flash 之间的交互。从实现步骤和代码示

例可以看出, LoadVars 方式具有以下优点:

- ✎ Flash 代码实现起来简单, 方便;
- ✎ 服务端接收页面和接收普通 Web 表单提交的数据一样处理, 因此服务器端程序编程简单, 而且适应多种编程技术, 如 ASP、JSP 等。

但也不难看出, 这种方式也具有明显的缺点, 如:

- ✎ 传递的变量不宜过多;
- ✎ 变量传递的值不宜过长;
- ✎ 变量传递值只能使用“字符串”这一种数据类型, 数据类型单一;
- ✎ 数据返回值当中不能有“&”字符, 因此比较复杂的返回值都需进行 URL 编码处理。

除了采用了 LoadVars 方式, 还可以采用以下多种方式实现 JSP 与 Flash 之间的交互。

① **Flash Remoting**: 这种方式支持数据类型比较多, 传递数据量比较大, 运行效率是现有几种方式当中最高的, 对各种后台的支持也比较好, 但是需要 Flash 端装 Flash Remoting MX 组件, 另外还需要后台服务端安装相应版本的 Flash Remoting 收费模块。

② **Web Service** 方法: 这种方法的通用性最好, 技术也比较成熟, 但目前开发工具的支持还不够, 开发难度较大。

③ **XMLSocket**: 可以实现与服务器端的直接连接。目前一些在线的 Flash 游戏, 多数采用此方式。但是在连接数量多的情况下对服务器的性能影响比较大, 且由于存在直接连接, 有时会遇到跨越防火墙的问题。

关于上述几种交互方式的具体实现, 可以查阅相关资料。

知识点索引

Flash; ActionScript; JSP; 数据库。

例程 1-7: 利用 JavaScript 脚本实现奥运倒计时日历

目的

- (1) 演示 JavaScript 与 JSP 之间的交互;
- (2) 日历组件的使用。

问题

如何在 Web 页面中显示一日历组件, 并实现奥运倒计时功能?

解决方案

利用 Calendar 组件来获取日历信息, 并通过 JSP 页面进行显示。利用 JavaScript 脚本实现与 Calendar 组件之间的动态交互。

实现步骤

- (1) 创建日历信息显示页面 canladar.jsp;
- (2) 创建奥运倒计时信息显示页面 olympic.jsp。

示例代码

本例程的所有示例代码均在 netbeans 的工程 jssample 内。

程序 1-18: canladar.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<!-- 显示任意年、月的日历，可选择不同的年、月。 -->
<%@ page import="java.util.*" %>
<%! String year;
      String month;
%>
<% month=request.getParameter("month");
   year =request.getParameter("year");
%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<title>日历</title>
<script Language="JavaScript">

function changeMonth()
{
    var mm="calendar.jsp?month="+document.sm.elements[0].selectedIndex+"&year="
+<%=year%>;
    window.open(mm,"_self");
}

}
</script>
</head>
<%! String days[]; %>
<%
days=new String[42];
for(int i=0;i<42;i++)
{
    days[i]="";
}
%>
<%
Calendar thisMonth=Calendar.getInstance();
if(month!=null&&(!month.equals("null")))
    thisMonth.set(Calendar.MONTH, Integer.parseInt(month));
if(year!=null&&(!year.equals("null")))
    thisMonth.set(Calendar.YEAR, Integer.parseInt(year));
year=String.valueOf(thisMonth.get(Calendar.YEAR));
month=String.valueOf(thisMonth.get(Calendar.MONTH));
thisMonth.setFirstDayOfWeek(Calendar.SUNDAY);
thisMonth.set(Calendar.DAY_OF_MONTH,1);
int firstIndex=thisMonth.get(Calendar.DAY_OF_WEEK)-1;
```



```

int maxIndex=thisMonth.getActualMaximum(Calendar.DAY_OF_MONTH);
for(int i=0;i<maxIndex;i++)
{
    days[firstIndex+i]=String.valueOf(i+1);
}
%>
<body>
<FORM name="sm" method="post" action="calendar.jsp">
    <%=year%>年 <%=Integer.parseInt(month)+1%>月
    <table border="0" width="168" height="81">
    <div align=center>
        <tr>
            <th width="25" height="16" bgcolor="#FFFF00"><font color="red">日</font>
        </th>
            <th width="25" height="16" bgcolor="#FFFF00">一</th>
            <th width="25" height="16" bgcolor="#FFFF00">二</th>
            <th width="25" height="16" bgcolor="#FFFF00">三</th>
            <th width="25" height="16" bgcolor="#FFFF00">四</th>
            <th width="25" height="16" bgcolor="#FFFF00">五</th>
            <th width="25" height="16" bgcolor="#FFFF00"><font color="green">六</font>
        </th>
    </tr>
    <% for(int j=0;j<6;j++) { %>
        <tr>
            <% for(int i=j*7;i<(j+1)*7;i++) { %>
                <td width="15%" height="16" bgcolor="#C0C0C0" valign="middle" align="center">
                    <a href="olympic.jsp?year=<%=year%>&month=<%=Integer.parseInt(month)+1%>
                    &date=<%=days[i]%>" target="main"><%=days[i]%></a></td>
                <% } %>
            </tr>
        <% } %>
    </div>
    </table>
    <table border="0" width="168" height="20">
        <tr>
            <td width=30%><select name="month" size="1" onchange="changeMonth()" >
                <option value="0">一月</option>
                <option value="1">二月</option>
                <option value="2">三月</option>
                <option value="3">四月</option>
                <option value="4">五月</option>
                <option value="5">六月</option>
                <option value="6">七月</option>
                <option value="7">八月</option>
                <option value="8">九月</option>
                <option value="9">十月</option>
                <option value="10">十一月</option>
                <option value="11">十二月</option>
            </select></td>
            <td width=28%><input type="text" name="year" value=<%=year%> size=4 maxlength=

```

```

=4></td>
<td>年</td>

<td width=28%><input type=submit value="查看"></td>
</tr>
</table>
</FORM>
<script Language="JavaScript">
    document.sm.month.options.selectedIndex=<%=month%>;
</script>
</body>
</html>

```

程序说明：程序通过日历组件 `Calendar` 来获取日历信息，并通过 `JavaScript` 脚本与客户进行交互，实现日历信息的刷新显示，以及倒计时页面的启动等操作。

程序 1-19: olympic.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>北京奥运开幕倒计时</title>
    </head>
    <body>
<%
String message="";
String year=request.getParameter("year");
String month=request.getParameter("month");
String day=request.getParameter("date");
Calendar olympic =Calendar.getInstance() ;
olympic.set(2008,8,8);
Calendar now =Calendar.getInstance() ;

now.set(Integer.parseInt(year),Integer.parseInt(month),Integer.parseInt(day));
if(now.after(olympic))message="很遗憾，北京奥运会早就开幕了";
else{
    int i;
    for( i=0;now.before(olympic);i++)now.add(Calendar.DATE,1);
    String s=Integer.toString(i);
    message="距离奥运开幕还有"+s+"天";
}
%>
<%=message%>

</body>
</html>

```

程序说明：获取传递来的请求信息，并据此来显示奥运倒计时信息，

运行结果

程序发布成功后,在浏览器地址栏输入“<http://localhost:8080/jssample/calendar.jsp>”,将得到如图 1-15 所示的运行结果。



图 1-15 显示动态日期

在日历上单击任意日期,将弹出新的窗口,显示距离北京奥运会开幕的日期,如图 1-16 所示。



图 1-16 显示奥运倒计时

讨论与思考

在 Web 应用开发中, JSP 和 JavaScript 分别是 Web 应用开发中的一种后端脚本技术和前端脚本技术, 后端 (JSP/Servlet) 和前端 (JavaScript) 是没法直接共用数据的, 但是两者有一个结合点: 即 HTML 页面。后端程序 (JSP) 把数据输出, 生成 HTML 页面到前端, 这时候生成的页面中的 JavaScript 代码才有可能得到所谓 JSP 的数据。同样的, 借助 HTML 中的超链接或表单, JavaScript 里的数据以 URL 参数或隐藏字段的形式提交给后端 JSP 代码, JSP 程序中即可得到 JavaScript 的数据。

将页面中的 JavaScript 数据提交给后台的 JSP 程序 最常用的是以下的方式。

① 将 JavaScript 的数据以 “xxx.JSP?var1=aaa&var2=bbb” 的形式作为 URL 的参数传给 JSP 程序, 此时在 JSP 中用 `<%String strVar1=request.getParameter("var1");%>` 就可以获取到 JavaScript 脚本传递过来的数据;

② 使用 JavaScript 通过在表单里加入隐藏域信息, 然后用表单提交的方式把数据传递给 JSP 程序。

页面中直接在 JavaScript 脚本中用 `<%=strVar1%>` 就可以把 JSP 程序中的数据传递给 JavaScript 脚本使用了。

JavaScript 使得 Web 页面的功能大大增强,并且可以完成一些智能化的操作,如用户输入校验等。在使用 JavaScript 时要注意避免过分使用 JavaScript。特别是不能将与业务逻辑相关的内容放置在 JavaScript 中,因为 JavaScript 是下载到客户端的浏览器上执行的,那样将会大大破坏程序的安全性,而且也破坏了 Web 应用开发的层次结构,因为 JavaScript 毕竟仅仅是表现层的一种技术。

知识点索引

JavaScript: 日历组件。

例程 1-8: 利用 XML、CSS 和 XSL 显示食谱信息

目的

演示如何利用 XML、XSL 和 CSS 技术构造动态 Web 页面。

问题

如何输出存放在 XML 文件中的食谱信息到 Web 浏览器?

解决方案 1: 利用 CSS 显示食谱 XML 文件的内容

创建一个 CSS 文件,对食谱 XML 文件中的每个节点定义一个显示样式。在食谱 XML 文件中通过 `<?xml-stylesheet type="text/css" href="simple.css"?>` 来引用 CSS 文件。

实现步骤

- (1) 创建显示食谱信息的样式表 `simple.css`。
- (2) 创建利用 CSS 显示的食谱文件 `simplecss.xml`。

示例代码

本例程的所有示例代码均在 netbeans 的工程 xml 内。

程序 1-20: `simple.css`

```
breakfast_menu
{
background-color: #ffffff;
width: 100%;
}
food
{
display: block;
margin-bottom: 30pt;
margin-left: 0;
}
name
{
```



```

color: #FF0000;
font-size: 20pt;
}
price, calories
{
color: #0000FF;
font-size: 20pt;
}
description
{
Display: block;
color: #000000;
margin-left: 20pt;
}

```

程序 1-21: simplecss.xml

```

<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/css" href="simple.css"?>
<breakfast_menu>
  <food>
    <name>芝麻蜂蜜包</name>
    <price>1.5 元</price>
    <description>芝麻蜂蜜包的馅用黑芝麻炒爆研碎拌适量蜂蜜制成，也可以加核桃仁、松子仁、甜杏仁、葡萄干等</description>
    <calories>650</calories>
  </food>
  <food>
    <name>纯精肉小笼包</name>
    <price>0.5 元</price>
    <description>精肉指用粮食、青草喂养的黑毛猪的肌肉。小笼包的制法如普通小笼包，但不能放味精。</description>
    <calories>900</calories>
  </food>
  <food>
    <name>绿豆红枣汤</name>
    <price>3 元</price>
    <description>清热，补肾</description>
    <calories>900</calories>
  </food>
  <food>
    <name>馒头</name>
    <price>0.3 元</price>
    <description>馒头用 100% 的全麦粉和酵母粉发酵制成，不得兑碱。没有全麦粉则用标准粉或荞麦粉代替</description>
    <calories>600</calories>
  </food>
</breakfast_menu>

```

运行结果

应用部署成功后，在浏览器的地址栏输入“http://localhost:8080/XML/simplecss.xml”，将得到如图 1-17 所示的运行结果。

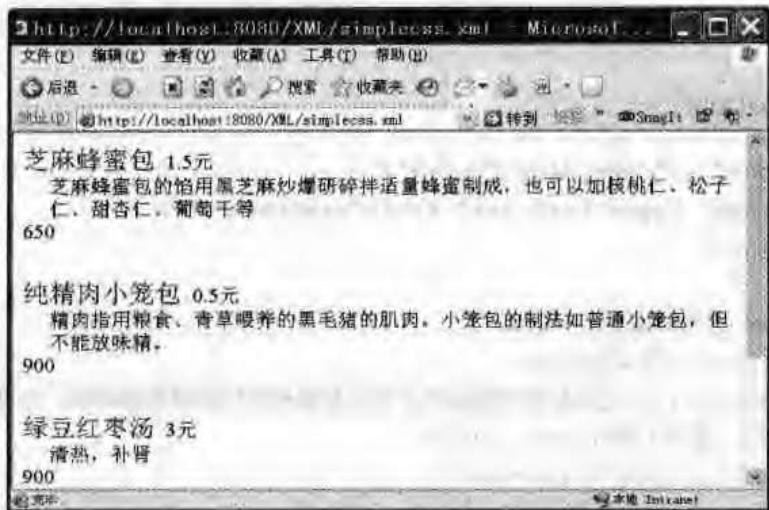


图 1-17 利用 CSS 显示 XML

解决方案 2：利用 XSL 显示食谱 XML 文件的内容

创建一个 XSL 文件，对食谱 XML 文件中的每个节点定义一个显示样式。在食谱 XML 文件中通过 `<?xml-stylesheet type="text/xsl" href="simple.xsl"?>` 来引用 XSL 文件。

实现步骤

- (1) 创建显示食谱信息的样式表 simple.xsl;
- (2) 创建利用 XSL 显示的食谱文件 simplexml.xml。

示例代码

程序 1-22: simple.xsl

```
<?xml version="1.0" encoding="utf-8"?>
<html xsl:version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org/TR/xhtml1/strict">
  <body style="font-family:Arial,Helvetica,sans-serif;font-size:12pt;
    background-color:#EEEEEE">
    <xsl:for-each select="breakfast_menu/food">
      <div style="background-color:teal;color:white;padding:4px">
        <span style="font-weight:bold;color:white">
          <xsl:value-of select="name"/></span>
          - <xsl:value-of select="price"/>
        </div>
        <div style="margin-left:20px;margin-bottom:1em;font-size:10pt">
          <xsl:value-of select="description"/>
        </div>
      </xsl:for-each>
    </body>
</html>
```



```

        <span style="font-style:italic">
            (<xsl:value-of select="calories"/> 卡路里)
        </span>
    </div>
</xsl:for-each>
</body>
</html>

```

程序 1-23: simplexml.xml

```

<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet type="text/xsl" href="simple.xml"?>
<breakfast_menu>
    <food>
        <name>芝麻蜂蜜包</name>
        <price>1.5 元</price>
        <description>芝麻蜂蜜包的馅用黑芝麻炒爆研碎拌适量蜂蜜制成，也可以加核桃仁、松子仁、甜杏仁、葡萄干等</description>
        <calories>650</calories>
    </food>
    <food>
        <name>纯精肉小笼包</name>
        <price>0.5 元</price>
        <description>精肉指用粮食、青草喂养的黑毛猪的肌肉。小笼包的制法如普通小笼包，但不能放味精。</description>
        <calories>900</calories>
    </food>
    <food>
        <name>绿豆红枣汤</name>
        <price>3 元</price>
        <description>清热，补肾</description>
        <calories>900</calories>
    </food>
    <food>
        <name>馒头</name>
        <price>0.3 元</price>
        <description>馒头用 100% 的全麦粉和酵母粉发酵制成，不得兑碱。没有全麦粉则用标准粉或荞麦粉代替</description>
        <calories>600</calories>
    </food>
</breakfast_menu>

```

运行结果

应用部署成功后，在浏览器的地址栏输入“<http://localhost:8080/XML/simplexml.xml>”，将得到如图 1-18 所示的运行结果。



图 1-18 利用 XSL 显示 XML

解决方案 3：利用 xalan 动态绑定 XML 和 XSL

xalan-java 是一套 XSLT 处理器，用来将 XML 文件根据 XSL 转换为 HTML、TEXT 和 XML 等其他类型文件格式，支持 XSLT1.0 和 XPATH 1.0 版。xalan-java 实现的是 transformation API for XML (TRaX) 接口，此接口已经成为 jaxp1.2 标准中的一部分。

运行准备

xalan 是多个 jar 包的集合。在开发程序之前，必须首先将 xalan 所需的 jar 包添加到工程的类路径下。xalan 的下载地址：<http://www.apache.org/dist/xml/xalan-j/>。

实现步骤

- (1) 创建显示 XML 文件的扩展样式表文件 simple2.xsl;
- (2) 创建文件 combine.jsp;
- (3) 使用 javax.xml.transform.TransformerFactory 的 newInstance 方法创建一个 javax.xml.transform.TransformerFactory 的实例;
- (4) 根据随机数获取 XSL 文件名;
- (5) 根据 XSL 文件名使用 javax.xml.transform.TransformerFactory 中的 newTemplates 方法创建一个 javax.xml.transform.Templates 的实例;
- (6) 使用 javax.xml.transform.Templates 的 newTransformer 方法创建一个 javax.xml.transform.Transformer 的实例;
- (7) 使用 Transformer 中的 transform 方法进行转换。其中，要进行转换的 XML 作为其第一个参数，JSP 的隐含对象 out 作为 transform 方法的第二个参数，确保转换后的结果输出到浏览器。

示例代码

程序 1-24: combine.jsp

```
<%@ page contentType="text/html; charset=UTF-8"
    import="java.io.File"
%>
```



```

<%
javax.xml.transform.TransformerFactory oFactory =
    javax.xml.transform.TransformerFactory.newInstance();
String xslname="";
int i = (int)(java.lang.Math.random()*10);
int r=i%2;
if(r==0) xslname="simple2.xsl";
else xslname="simple.xsl";

javax.xml.transform.Templates oTemplates = oFactory.newTemplates(
    new javax.xml.transform.stream.StreamSource(request.getRealPath("/")
    +xslname)
    );

javax.xml.transform.Transformer transformer = oTemplates.newTransformer();

transformer.transform
(
    new javax.xml.transform.stream.StreamSource(request.getRealPath("/")
    +"simplexml.xml"),
    new javax.xml.transform.stream.StreamResult( out ));
%>

```

运行结果

应用部署成功后，在浏览器的地址栏输入“<http://localhost:8080/XML/combine.jsp>”，将得到如图 1-19 所示的运行结果。

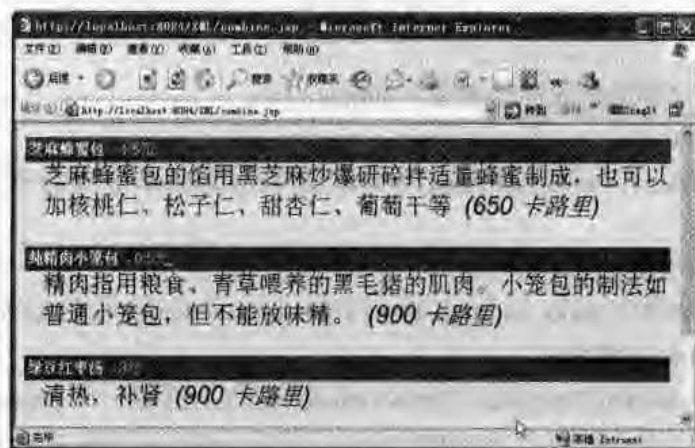


图 1-19 利用 xalan 动态绑定 XML 和 XSL

讨论与思考

XML 由于其良好的可扩展性和跨平台特性，自它诞生起，就成为 Java EE 的宠儿。区别于注重表现的 HTML，XML 的最大特点是内容与表现形式是分离的，在一个 XML 的源文件中并没有关于它表现形式的信息。XML 的最大特点就是揭示了信息本身的含义，用于自动化的电子文档交换是最优的，如果一个 XML 文件仅仅用于交换信息，就无须考虑它的显示问题。因此，XML 成为信息集成中的桥梁，也担

负起 Web Service 的重任。

编辑 XML 文件，开发人员仅仅需要关注文件的内容、信息的结构，至于它怎么显示，则交给 CSS（叠层样式表）和 XSL（可扩展样式语言）来完成。这就使得用户可以根据需要来定义数据的表现形式。在 XML 文件中，使用的基本上是自定义的标记，显然一个浏览器是无法理解这些标记的，目前浏览器仅仅是作为一个 XML 文件的解析器——只要 XML 文件是格式良好的，那么它就将文件原封不动地显示出来。

对于同一个 XML 文件，如果赋予它不同的 CSS，那么它就有不同的显示效果。

XSL 也是一种显示 XML 文件的规范。与 CSS 不同的是，XSL 是遵循 XML 的规范来制定的。也就是说，XSL 文件本身符合 XML 的语法规则。XSL 在排版样式的功能上要比 CSS 强大。比如：CSS 适用于那些元素顺序不变的文件，它不能改变 XML 文件中元素的顺序——元素在 XML 文件中是按什么顺序排列的，那么通过 CSS 表现出来顺序不能改变。对于那些需要经常按不同元素排序的文件，就要使用 XSL。

xalan 作为一个良好的工具，使得根据业务逻辑动态控制 XML 文件的显示格式甚至输出格式成为可能。

但是使用 XML+XSL 替代 HTML 作为 Web 前端显示解决方案的时机尚未成熟，因为目前无论是开发技术还是开发工具都还不够完善，而且 XML 在显示前需要经过复杂的运算处理，这对性能和响应速度提出了新的挑战。

知识点索引

XML; XSL; CSS; xalan。

例程 1-9：利用标准标记库显示本地化信息

目的

- (1) 演示标准标记库的使用；
- (2) 信息的本地化显示。

问题

网站中设计的网页经常需要被不同国家和地区的人员访问。一些特殊信息如数字、日期、货币等具有明显的本地化特征，那么如何显示页面中的这些本地化信息？

解决方案

从 `HttpRequest` 请求中获取区域属性，然后使用标准标记库的 `<fmt>` 标签来实现信息本地化显示。

知识链接

标准标记库：为了提高 JSP 页面的开发效率，JSP2.0 版本推出了 JSTL 标准标记库来帮助开发人员完成一些常见的开发任务。实现这些常见任务的标记称为标准标记。这些常见任务包括迭代和条件判断、数据管理格式化、XML 操作以及数据库访问等。根据标准标记的内容，JSTL 可以分为 5 部分，如表 1-3 所示。

表 1-3 JSTL 的功能标记库

标记库名称	URI	前 缀	说 明
core	http://java.sun.com/jsp/jstl/core	c	核心功能实现, 包括变量管理、迭代和条件判断等
fn	http://java.sun.com/jsp/jstl/fmt	fmt	国际化, 数据格式显示
SQL	http://java.sun.com/jsp/jstl/sql	sql	操作数据库
XML	http://java.sun.com/jsp/jstl/xml	x	操作 XML
Fn	http://java.sun.com/jsp/jstl/functions	fn	常用函数库, 包括 String 操作、集合类型操作等

示例代码

本例程的所有示例代码均在 netbeans 的工程 jstlsample 内。

程序 1-25: fmt.jsp

```
<%@ taglib uri="http://java.sun.com/jstl/core_rt" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jstl/fmt_rt" prefix="fmt" %>
<%@page pageEncoding="UTF-8"%>
<html>
<head><title>本地化显示数字和日期</title></head>
<body>
<h2>数字和日期的本地化显示</h2>

<%-- create an object representing the current date --%>
<jsp:useBean id="now" class="java.util.Date"/>

<%-- set the locale to German language, Swiss country code --%>
<fmt:setLocale value="${requestScope[Local]}" />
<strong>日期显示:</strong>
<%-- output the date --%>
<fmt:formatDate type="both" value="${now}" dateStyle="full" timeStyle="short" />
<br />

<strong>数字显示:</strong>

<%-- output the equivalent of java.util.Date.getTime() to show how numbers are
formatted
--%>
<fmt:formatNumber value="${now.time}" /><br />
<strong>货币显示:</strong>
<fmt:formatNumber type="currency" value="10000"></fmt:formatNumber>
<br /><br />
</body>
</html>
```

运行准备

在运行前需要将标准标记库所需的 jar 文件添加到项目的 classpath 下。具体操作步骤请参考例程 1-3 的实现步骤中的步骤 (1)。

运行结果

应用部署成功后,在浏览器的地址栏输入“`http://localhost:8080/jstlsample/fmt.jsp`”,将得到如图 1-20 所示的运行结果。



图 1-20 信息的本地化显示

讨论与思考

日期,数字等信息的本地化显示是开发真实的企业应用中不可回避的一个问题,JSTL 为实现日期、数字等信息的本地化提供了一个便捷的解决方案。关于本地化问题,在例程 5-3 中还会进行研究。

知识点索引

JSTL: 本地化。

例程 1-10: 在 Web 页面中引入版权信息声明

目的

演示如何在 Web 页面(包括 JSP 页面、Servlet)中引入其他页面的内容

问题

在网站的开发过程中,为了维护网站的合法权益,一般在网站的所有页面上都加上版权声明。在每个页面的开发过程中都直接嵌入代表版权信息的代码显然是项很繁杂的工作,而且也不易维护。那么如何解决这一问题?

解决方案 1: 使用 include 指令在 JSP 中包含版权信息

使用 JSP 指令 include 将版权信息页面包含到 JSP 页面中。

知识链接

JSP 指令是从 JSP 向 Web 容器发送的消息,它用来设置页面的全局属性,如输出内容类型等。指令

不向客户端输出任何内容，其作用范围仅限于包含指令本身的 JSP 页面。

JSP 的指令格式为：

```
<%@ 指令名 属性="属性值"%>
```

指令名有 `page`、`include` 和 `taglib` 三种。`taglib` 指令允许页面使用扩展标记：`page` 指令用来定义整个 JSP 页面的全局属性，如 `import`、`contentType`、`isThreadSafe` 和 `language` 等；`include` 指令向 JSP 页面内某处嵌入一个文件，这个文件可以是 HTML 文件、JSP 文件或其他文本文件。需要着重说明的是：通过 `include` 指令包含的文件是由 JSP 分析的，并且这部分分析工作是在转换阶段（JSP 文件被编译为 Servlet 时）进行的。

实现步骤

- (1) 创建包含版权信息的页面 `Copyright.jsp`；
- (2) 创建页面 `c1.jsp`，在其中使用指令 `include` 包含页面 `copyright.jsp`。

示例代码

本例程的所有示例代码均在 `netbeans` 的工程 `copyright` 内。

程序 1-26: `Copyright.jsp`

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
Copyright &copy; 郝玉龙 All Rights Reserved
```

程序 1-27: `c2.jsp`

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>include 指令包含版权信息</title>
  </head>
  <body>
    <h1>欢迎访问我们的网站.....</h1>

    <%@ include file="Copyright.jsp" %>
  </body>
</html>
```

运行结果

程序运行结果如图 1-21 所示。



图 1-21 使用 include 指令包含版权信息

解决方案 2: 使用动作组件在 JSP 中包含版权信息

知识链接

JSP 动作组件是一些 XML 语法格式的标记, 用来控制 Web 容器的行为。利用 JSP 动作组件可以动态地向页面中插入文件、重用 JavaBean 组件、把用户重定向到另外的页面等。常见的 JSP 动作组件共有以下几种。

<jsp:include>: 在页面被请求的时候引入一个文件。

<jsp:param>: 在动作组件中引入参数信息。

<jsp:forward>: 把请求转到一个新的页面。

<jsp:setProperty>: 设置 JavaBean 的属性。

<jsp:getProperty>: 输出某个 JavaBean 的属性。

<jsp:useBean>: 寻找或者实例化一个 JavaBean。

其中 include 动作组件把指定文件插入正在生成的页面。其语法如下:

```
<jsp:include page="文件名" flush="true"/>。
```

参数 flush 为 true 表示清除以前对此包含页面的缓存内容。

实现步骤

- (1) 创建包含版权信息的页面 Copyright.jsp;
- (2) 创建页面 c2.jsp, 在其中使用动作组件 include 包含页面 copyright.jsp。

示例代码

程序 1-28: c2.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>include 动作组件包含版权信息</title>
```



```

</head>
<body>
<h1>欢迎访问我们的网站.....</h1>

<jsp:include page="Copyright.jsp" flush="true"/>

</body>
</html>

```

运行结果

程序运行结果如图 1-22 所示。



图 1-22 使用 include 动作组件包含版权信息

解决方案 3：使用标准标记<c:import>在 JSP 中包含版权信息

实现步骤

(1) 将 JSTL 库文件添加到工程的 classpath 路径下。具体操作为：在【项目】视图选中工程的库文件夹，单击右键，在弹出的快捷菜单中选择【添加库】选项弹出【添加库】对话框，如图 1-23 所示。



图 1-23 【添加库】对话框

(2) 在列表中选择【JSTL 1.1】，单击【添加库】按钮，则将 JSTL 库成功地添加到项目的 classpath 下。单击【库】文件夹，可以看到新增的 JSTL 库，如图 1-24 所示。



图 1-24 新增加的库

(3) 利用 `taglib` 指令将标准标记引入到 JSP 页面。示例代码如下：

```
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
```

(4) 使用 `<c:import>` 指令将版权信息包含到页面中。

示例代码

程序 1-29: c3.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>使用标准标记库导入版权信息</title>
    </head>
    <body>
        <h1>欢迎访问我们的网站.....</h1>
        <c:import url="Copyright.jsp" />
    </body>
</html>
```

运行结果

程序运行结果如图 1-25 所示。



图 1-25 使用标准标记库导入版权信息

解决方案 4: 在 Servlet 响应中包含版权信息

利用 `RequestDispatcher` 对象的 `include` 方法将其他 Web 组件对当前请求的响应包含到 Servlet 的响应中。

实现步骤

- (1) 创建一个 Servlet `IncludeServlet`;
- (2) 调用 `request` 对象的 `getRequestDispatcher` 获取 `RequestDispatcher` 对象;
- (3) 调用 `RequestDispatcher` 对象的 `include()` 将其他版权信息包含到 Servlet 的响应中。

示例代码

程序 1-30: `IncludeServlet.java`

```
import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class IncludeServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse
    response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet 中包含版权信息" +
            "</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>欢迎来到我们的网站.....</h1><br><br>");
        RequestDispatcher dispatcher = request.getRequestDispatcher
            ("Copyright.jsp");
        dispatcher.include(request, response);
        out.println("</body>");
        out.println("</html>");
        out.close();

        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }
}
```

```

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}
}

```

运行结果

程序运行结果如图 1-26 所示。



图 1-26 在 Servlet 响应中包含版权信息

讨论与思考

通过组件之间的包含机制来集成不同 Web 组件的功能，共同完成复杂的任务是开发中经常遇到的问题。在上面的包含版权信息的示例中，一共列举的 4 种解决方案来实现 Java EE 技术框架下 Web 组件之间的包含。

首先对 `include` 动作组件和 `include` 指令元素实现的包含机制进行深入讨论。`include` 指令实现的包含属于静态包含，它是在 JSP 页面编译成 Servlet 之前，将被包含的页面代码加入到 JSP 页面代码中形成一个新的 JSP 文件，然后编译成 Servlet 文件，来响应客户端的请求。在这种机制下，相当于在本地生成了一个被包含文件的副本，因此，除非重新编译，否则，被包含文件的更新是不能够反映到包含组件的最终显示结果中。

`include` 动作组件属于动态包含，即动作组件在执行时才对被包含的文件进行处理，将被包含的组件的输出结果集成到页面自身的输出中。因此 JSP 页面和它所包含的文件在逻辑上和语法上是独立的，如果对被包含的文件进行了修改，那么修改后的更新可以立即反映到包含文件的显示中。

因此，从上面的分析可以看出，为了提高应用程序的性能，`include` 动作组件适合包含频繁动态更新的内容，而 `include` 指令适合包含变动较少的静态内容。

JSTL 标准标记库可以看作 JSP 动作组件的一种扩展，因此，利用 `<c:import>` 包含外部文件也属于动态包含的一种，不过，`<c:import>` 的扩展性要灵活得多，不仅可以包含本 Web 应用的资源，还可以包含其他 Web 应用的资源，甚至其他服务器上的 Web 资源。详细的使用步骤可以参阅相关资料。

在 Servlet 的响应中，包含其他资源对请求响应的方法是：获取被包含资源的 `RequestDispatcher` 对象，然后调用 `RequestDispatcher` 对象的 `include` 方法在响应中包含被包含资源的响应，因此也属于

动态包含的范畴。

知识点索引

include 指令; include 动作组件; JSTL; RequestDispatcher。

例程 1-11: 实现带图形验证码的用户登录

目的

- (1) 创建 Java 图片对象;
- (2) 输出非 HTML 文件到客户端;
- (3) 客户端提交参数的处理;
- (4) 会话信息管理。

问题

如何在登录界面提供图形验证码, 以防止对登录系统进行暴力破解?

解决方案

首先在服务器端生成一组随机数字作为验证码存储在会话中, 然后生成包含验证码的图像文件并发送到客户端浏览器, 最后将客户端提交的验证码与会话中存储的信息进行对比, 来确认客户端提交信息正确与否。

示例代码

本例程的所有示例代码均在 netbeans 的工程 IDCode 内。

程序 1-31: index.html

```
<%@ page contentType="text/html; charset=gb2312" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>带验证码的登录</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">
<META HTTP-EQUIV="Expires" CONTENT="0">
</head>
<body>
<form method="post" action="check.jsp">
<table>
<tr>
<td align="left">用户名: </td>
<td><input type="text" name="user" value="" /></td>
</tr>
```

```

<tr>
    <tr>
    <td align=left>密 码: </td>
        <td><input type="password" name="password" value="" /></td>
    </tr>
    <td align=left>认证码: </td>
    <td><input type=text name=rand maxlength=4 value=""></td>
    </tr>

    <tr>
    <td colspan=2 align=center><input type=submit value="提交"></td>
    </tr>
</form>
</body>
</html>

```

程序说明：主要提供用户登录界面。对于图形验证码的实现很简单，只需要在页面中添加一个标记即可，其中标记的 src 属性指向一个专门用来生成图像的 JSP 页面。

程序 1-32: image.jsp

```

<%@ page contentType="image/jpeg" import="java.awt.*,
java.awt.image.*,java.util.*,javax.imageio.*" %>
<%!
Color getRandColor(int fc,int bc){
    Random random = new Random();
    if(fc>255) fc=255;
    if(bc>255) bc=255;
    int r=fc+random.nextInt(bc-fc);
    int g=fc+random.nextInt(bc-fc);
    int b=fc+random.nextInt(bc-fc);
    return new Color(r,g,b);
}

%>
<%

response.setHeader("Pragma","No-cache");
response.setHeader("Cache-Control","no-cache");
response.setDateHeader("Expires", 0);
int width=60, height=20;
BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);
Graphics g = image.getGraphics();
Random random = new Random();
g.setColor(getRandColor(200,250));
g.fillRect(0, 0, width, height);
g.setFont(new Font("Times New Roman",Font.PLAIN,18));
g.setColor(getRandColor(160,200));
for (int i=0;i<155;i++)
{
    int x = random.nextInt(width);
    int y = random.nextInt(height);

```



```

        int x1 = random.nextInt(12);
        int y1 = random.nextInt(12);
        g.drawLine(x, y, x+x1, y+y1);
    }
    String sRand="";
    for (int i=0;i<4;i++){
        String rand=String.valueOf(random.nextInt(10));
        sRand+=rand;

        g.setColor(new Color(20+random.nextInt(110),20+random.nextInt(110),20+random.
            nextInt(110)));

        g.drawString(rand,13*i+6,16);
    }
    session.setAttribute("rand",sRand);
    g.dispose();
    ImageIO.write(image, "JPEG", response.getOutputStream());
    %>

```

程序说明：用来生成随机图片并输出到客户端。首先生成随机颜色的图片背景，然后随机生成 4 个数字作为验证码，调用 Graphics 对象的 drawString() 方法将验证码叠加到背景图片上。最后将验证码信息放到 Session 对象中以便进行后面的验证操作，并通过调用 ImageIO 的 write() 方法写图片输出到 response 对象的输出流中。

程序 1-33: check.jsp

```

<%@ page contentType="text/html; charset=gb2312" language="java"
import="java.sql.*" errorPage="" %>
<html>
<head>
<title>带验证码的登录验证页面</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">
<META HTTP-EQUIV="Expires" CONTENT="0">
</head>

<body>
<%
    String rand = (String)session.getAttribute("rand");
    String input = request.getParameter("rand");
    %>
<%
    if (rand.equals(input)) {
        String user = request.getParameter("user");
        String password=request.getParameter("password");
    %>
    <font color=green>验证码输入成功! <br>
    您登录的用户名是<%=user%><br>
    您登录的密码是<%=password%>
    </font>

```

```
<%  
    } else {  
%>  
<font color=red>验证码错误或已经失效....</font>  
<%  
    }  
%>  
</body>  
</html>
```

程序说明：按照标准流程，程序需要连接数据库，根据从数据库获取的信息对用户登录进行验证。为代码简洁，程序仅用来实现登录时图形验证码的校验。首先调用 `getParameter("rand")` 方法获取输入的验证码，然后调用 `session` 对象的 `getAttribute("rand")` 方法获取存储的验证码的值，并根据二者对比的结果进行正确处理。

运行结果

程序发布成功后，在浏览器地址栏输入“`http://localhost:8080/IDCode/`”，将得到如图 1-27 所示的运行结果。



图 1-27 带图形验证码的用户登录界面

输入登录的用户名与密码，按照图形验证码的提示信息输入验证码，单击【提交】按钮，将得到如图 1-28 所示的运行结果。

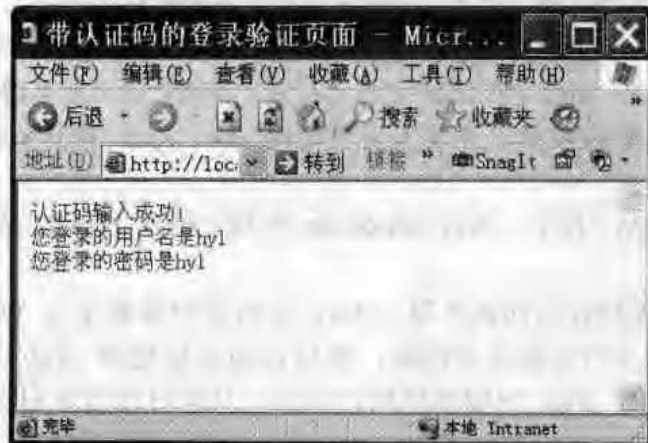


图 1-28 登录成功提示信息

讨论与思考

在 Web 页面的展示中, 图像的运用也是一个很重要的技巧, 特别是一些特殊信息如验证码、数学公式、统计曲线图表的展示, 限于 HTML 标记的局限性, 可以采用在服务器端生成图片, 然后发送到客户端浏览器。

知识点索引

Java 图像处理; 发送非 HTML 文档; 会话处理; 安全。

例程 1-12: 利用 Ajax 实现网上智能订餐

目的

演示如何利用 Ajax 技术实现智能用户界面。

问题

在网上订餐程序中, 要求实现智能订餐的功能。具体要求如下:

- (1) 顾客订餐信息自动更新在已点菜目列表中;
- (2) 自动计算餐费, 并在餐费超出一定数目时具有自动提示功能;
- (3) 自动计算食物热量总和, 并在热量超出一定数目时具有自动提示功能。

解决方案

利用 Ajax 技术实现已点菜目列表, 这样就能够确保顾客点餐信息的及时快速更新。利用 JavaScript 脚本编程, 实现点餐时的自动提醒功能。

知识链接

Ajax 是 Asynchronous JavaScript and XML (以及动态 HTML 等) 的英文缩写。Ajax 从本质上讲不是一门新技术, 而是一个新的技术体系框架。它主要采用以下基本技术:

- ① HTML 用于建立 Web 表单并确定应用程序其他部分使用的字段;
- ② JavaScript 代码是运行 Ajax 应用程序的核心代码, 帮助改进与服务器应用程序的通信;
- ③ DHTML 用于动态更新表单, 开发人员将使用 div、span 和其他动态 HTML 元素来标记 HTML;
- ④ 文档对象模型 (DOM) 用于 (通过 JavaScript 代码) 处理 HTML 结构和 (某些情况下) 服务器返回的 XML。

作为一种 Web 应用程序开发的新手段, Ajax 采用客户端脚本与 Web 服务器交换数据。所以, 不必采用导致中断交互的完整页面刷新, 就可以动态地更新 Web 页面。使用 Ajax, 可以创建更加丰富、更加动态的 Web 应用程序用户界面, 其即时性与可用性甚至能够接近本机桌面应用程序。

实现步骤

(1) 编写 JavaScript 脚本 ajax1, 获取 XMLHttpRequest 对象以便与后台服务器程序进行通信, 以及设置处理服务器返回 XML 文档的方法。

(2) 编写 JavaScript 脚本 dish2, 利用 XMLHttpRequest 对象实现与后台服务器应用程序的交互以及前端页面的刷新显示。

(3) 生成代表点餐程序中抽象对象的 JavaBean。示例中共有三个 JavaBean: 代表菜单对象的 Catalog, 代表食物对象的 Item 和代表已点菜目对象的 dish, 它们都存储在 com.example.ajax.util 包中。

(4) 编写 Servlet dishServlet 实现对 JavaScript 脚本请求的处理。

(5) 编写页面 index.jsp, 用来实现点餐程序与终端用户的交互和展示。

示例代码

本例程的所有示例代码均在 netbeans 的工程 Ajax 内。

程序 1-34: ajax1.js

```
function newXMLHttpRequest() {

    var xmlreq = false;

    // Create XMLHttpRequest object in non-Microsoft browsers
    if (window.XMLHttpRequest) {
        xmlreq = new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        try {
            // Try to create XMLHttpRequest in later versions
            // of Internet Explorer
            xmlreq = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e1) {
            // Failed to create required ActiveXObject

            try {
                // Try version supported by older versions
                // of Internet Explorer

                xmlreq = new ActiveXObject("Microsoft.XMLHTTP");
            } catch (e2) {

                // 生成对象失败
                xmlreq = false;
            }
        }
    }

    return xmlreq;
}

function getReadyStateHandler(req, responseXmlHandler) {
```



```

return function () {
    //检查请求状态是否完成
    if (req.readyState == 4) {
        // 检查是否从服务器得到成功响应
        if (req.status == 200) {
            // 将响应 XML 文件传递给指定的处理方法
            responseXmlHandler(req.responseXML);
        } else {
            // 意外处理
            alert("HTTP error "+req.status+": "+req.statusText);
        }
    }
}
}

```

程序说明：程序用来实现 Ajax 框架的基本操作，其中方法 `newXMLHttpRequest()` 用来获取 `XMLHttpRequest` 对象，由于不同的浏览器创建 `XMLHttpRequest` 对象的方法是不同的，代码中使用了多种方法力图避免浏览器的差异对创建 `XMLHttpRequest` 的影响，但仍旧存在 `XMLHttpRequest` 对象创建失败的可能，此时方法将返回 `false`。方法 `getReadyStateHandler()` 用来设定处理响应返回的 XML 文档的方法。在将响应 XML 传递给指定的回调方法之前，它将检查请求的状态以便对请求处理过程中的意外进行处理。

程序 1-35: dish2.js

```

var lastdishUpdate = 0;
function addToDish(itemCode) {

    var req = newXMLHttpRequest();

    req.onreadystatechange = getReadyStateHandler(req, updateDish);

    req.open("POST", "dish.do", true);
    req.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    req.send("action=add&item="+itemCode);
}

function emptyDish() {

    var req = newXMLHttpRequest();

    req.onreadystatechange = getReadyStateHandler(req, updateDish);

    req.open("POST", "dish.do", true);
    req.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
    req.send("action=clear");
}

function updateDish(dishXML) {
    var dish = dishXML.getElementsByTagName("dish")[0];
}

```

```

var generated = dish.getAttribute("generated");
if (generated > lastdishUpdate) {
    lastdishUpdate = generated;
    var contents = document.getElementById("contents");
    contents.innerHTML = "";
    var items = dish.getElementsByTagName("item");
    for (var i = 0 ; i < items.length ; i++) {
        var item = items[i];
        var name = item.getElementsByTagName("name")[0].firstChild.nodeValue;
        var quantity = item.getElementsByTagName("quantity")[0].firstChild.nodeValue;
        var listItem = document.createElement("li");
        listItem.appendChild(document.createTextNode(name+" x "+quantity));
        contents.appendChild(listItem);
    }
}

document.getElementById("total").innerHTML = dish.getAttribute("total");
var money =dish.getAttribute("total");
money=money.substring(0,money.length-3);
if(money > 30)alert("您的午餐已经超过 30 元! 注意节约啊,:)");
document.getElementById("heat").innerHTML = dish.getAttribute("heat");
var h =dish.getAttribute("heat");
h=h.substring(0,h.length-1);
if(h > 3000)alert("您的午餐热量已经超标! 注意保持体形啊,:)");
}

```

程序说明：程序用来实现与服务器端的交互。

addToDish(itemCode)方法用来响应客户点餐请求，并将请求发送到服务器。它的实现步骤如下。

- ① 调用 newXMLHttpRequest()方法创建一个 XMLHttpRequest 对象 req。
- ② 调用 getReadyStateHandler(req, updateDish) 方法设置 XMLHttpRequest 对象 req 的回调方法为 updateDish()。

③ 调用 open("POST", "dish.do", true) 方法建立与服务器的连接。

④ 调用 setRequestHeader()方法设置请求的头部信息。

⑤ 调用 send()方法将请求信息发送到服务器程序。

updateDish(dishXML)方法用来处理服务器返回的响应 XML 文件，并对页面进行刷新，其中参数 dishXML 为服务器返回的响应 XML 文件。它的实现步骤如下。

① 调用 dishXML 的 getElementsByTagName()方法获取 XML 文件中指定的内容 dish。

② 调用 dish 的 getAttribute("generated")方法获取响应产生的时间并与保存的时间戳进行对比，如果响应产生时间新于保存的时间戳，则用新的响应时间取代保存时间戳的内容。注意：这里进行时间戳对比操作是 Ajax 处理特有的，因为在 Ajax 框架下，脚本与服务器是一种异步通信方式，即脚本请求发送到服务器是不等待服务器的响应就返回，这样就造成服务器对前端请求的响应“后发而先至”的现象。为了避免上面的现象，必须进行响应时间戳的对比。

③ 对 dish 的内容进行处理生成 HTML 代码，并调用 appendChild()方法添加到前端页面显示。

④ 对 dish 的内容进行处理实现对餐费的自动提醒功能。

⑤ 对 dish 的内容进行处理实现对实物热量的自动提醒功能。

emptyDish()方法用来向服务器发出请求，清空已选的食物信息。

程序 1-36: DishServlet.java

```

package com.example.ajax.servlet;

import com.example.ajax.util.Dish;
import javax.servlet.http.*;

import java.util.Enumeration;

public class DishServlet extends HttpServlet {
    public void doPost(HttpServletRequest req, HttpServletResponse res) throws
        java.io.IOException {
        Enumeration headers = req.getHeaderNames();
        while (headers.hasMoreElements()) {
            String header =(String) headers.nextElement();
            System.out.println(header+": "+req.getHeader(header));
        }
        Dish dish = getDishFromSession(req);
        String action = req.getParameter("action");
        if ("clear".equals(action)) dish.clear();
        String item = req.getParameter("item");

        if ((action != null)&&(item != null)) {
            if ("add".equals(action)) {
                dish.addItem(item);
            }
        }
        String dishXml = dish.toXml();
        res.setContentType("text/xml;charset=UTF-8");
        res.getWriter().write(dishXml);
    }

    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
        java.io.IOException {
        doPost(req,res);
    }

    private Dish getDishFromSession(HttpServletRequest req) {
        HttpSession session = req.getSession(true);
        Dish dish = (Dish)session.getAttribute("dish");
        if (dish == null) {
            dish = new Dish();
            session.setAttribute("dish", dish);
        }
        return dish;
    }
}

```

程序说明：对前端脚本发送来的请求进行处理。虽然 Servlet 处理的是脚本发送来的信息，从实现代码可以看到，与处理普通的 HTTP 请求信息没什么不同，唯一的不同之处在于对客户端的响应上。在 Ajax

中,对客户端的响应必须以 XML 的形式返回,因此在 doPost() 方法中,调用以下代码来设置对前端请求的响应类型。

```
String dishXml = dish.toXml();res.setContentType("text/xml;charset=UTF-8");
res.getWriter().write(dishXml);
```

注意:如果响应中包含汉字信息,则设置响应类型时必须将字符集设置为 UTF-8,即如本例所示,否则将会有乱码出现。

程序 1-37: Catalog.java

```
package com.example.ajax.util;

import java.util.*;

public class Catalog {
    private static Map<String,Item> items;
    static {
        items = new HashMap<String,Item>();
        items.put("c1",new Item("c1","冬瓜炖排骨","温虚补肾",1500,1200));
        items.put("c2",new Item("c2","蚝油生菜","清淡爽口",800,500));
        items.put("c3",new Item("c3","家常豆腐","老少皆宜",1200,1300));
        items.put("c4",new Item("c4","可乐鸡翅","别有风味",1800,3500));
        items.put("z1",new Item("z1","肉夹馍","陕西风味",200,1200));
        items.put("z2",new Item("z2","兰州拉面","微辣",600,1700));
        items.put("z3",new Item("z3","饺子","素三鲜的",2000,1600));
        items.put("t1",new Item("t1","西红柿鸡蛋","新鲜爽口",500,900));
        items.put("t2",new Item("t2","冬瓜海带","清淡解油",400,300));
        items.put("t3",new Item("t3","皮蛋瘦肉粥","营养丰富",900,1500));
    }

    public Collection<Item> getAllItems() {
        return items.values();
    }

    public boolean containsItem(String itemCode) {
        return items.containsKey(itemCode);
    }

    public Item getItem(String itemCode) {
        return items.get(itemCode);
    }
}
```

程序说明:代表网上提供的各种食物信息列表。代码中使用了泛型的概念。

程序 1-38: Dish.java

```
package com.example.ajax.util;

import java.math.BigDecimal;
import java.util.*;
```



```

public class Dish {

    private HashMap<Item,Integer> contents;

    public Dish() {
        contents = new HashMap<Item,Integer>();
    }

    public void addItem(String itemCode) {
        Catalog catalog = new Catalog();
        if (catalog.containsItem(itemCode)) {
            Item item = catalog.getItem(itemCode);
            int newQuantity = 1;
            if (contents.containsKey(item)) {
                Integer currentQuantity = contents.get(item);
                newQuantity += currentQuantity.intValue();
            }
            contents.put(item, new Integer(newQuantity));
        }
    }

    public void clear() {
        contents.clear();
    }

    public String toXml() {
        StringBuffer xml = new StringBuffer();
        xml.append("<?xml version='1.0' encoding='UTF-8'>\n");
        xml.append("<dish generated='"+System.currentTimeMillis()+"' "
            + "heat='"+getHeatTotal()+"' total='"+getPriceTotal()+"'\n");

        for (Iterator<Item> I = contents.keySet().iterator(); I.hasNext(); ) {
            Item item = I.next();
            int itemQuantity = contents.get(item).intValue();
            xml.append("<item code='"+item.getCode()+"'\n");
            xml.append("<name>");
            xml.append(item.getName());
            xml.append("</name>\n");
            xml.append("<quantity>");
            xml.append(itemQuantity);
            xml.append("</quantity>\n");
            xml.append("</item>\n");
        }

        xml.append("</dish>\n");
        return xml.toString();
    }

    private String getPriceTotal() {
        int total = 0;
        for (Iterator<Item> I = contents.keySet().iterator(); I.hasNext(); ) {
            Item item = I.next();
            int itemQuantity = contents.get(item).intValue();
            total += (item.getPrice() * itemQuantity);
        }
    }
}

```

```

        return new BigDecimal(total).movePointLeft(2)+"元";
    }
    private String getHeatTotal() {
        int totalHeat = 0;
        for (Iterator<Item> I = contents.keySet().iterator(); I.hasNext(); ) {
            Item item = I.next();
            int itemQuantity = contents.get(item).intValue();
            totalHeat += (item.getHeat() * itemQuantity);
        }
        return new BigDecimal(totalHeat)+"卡";
    }
}

```

程序说明：代表顾客的点餐信息，其中值得注意的是方法 toXml()，它将 JavaBean 的信息转换为一个 XML 文件，以便返回给前端的脚本。

程序 1-39: Item.java

```

package com.example.ajax.util;

import java.math.BigDecimal;

public class Item {
    private String code;
    private String name;
    private String description;
    private int price;
    private int heat;
    public Item(String code,String name,String description,int price,int heat) {
        this.code=code;
        this.name=name;
        this.description=description;
        this.price=price;
        this.heat=heat;
    }
    public String getCode() {
        return code;
    }
    public String getName() {
        return name;
    }
    public String getDescription() {
        return description;
    }
    public int getPrice() {
        return price;
    }

    public String getFormattedPrice() {
        return new BigDecimal(price).movePointLeft(2)+"元";
    }
}

```



```

public boolean equals(Object o) {
    if (this == o) return true;
    if (this == null) return false;
    if (!(o instanceof Item)) return false;
    return ((Item)o).getCode().equals(this.code);
}

public int getHeat() {
    return heat;
}

public String getFormattedHeat(){
    return Integer.toString(heat)+"卡";
}

public void setHeat(int heat) {
    this.heat = heat;
}
}

```

程序说明：代表食物信息，共有 5 个属性。

程序 1-40: index.jsp

```

<%@ page import="java.util.*" %>
<%@page pageEncoding="UTF-8"%>
<%@ page import="com.example.ajax.util.*" %>
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<script type="text/javascript" language="javascript" src="ajax1.js"></script>
<script type="text/javascript" language="javascript" src="dish2.js"></script>
</head>
<body>
<div style="float: left; width: 500px">
<h2>今日菜谱</h2>
<table border="1">
    <thead><th>名称</th><th>描述</th><th>热量</th><th>价格</th><th></th></thead>
    <tbody>
        <%
            for (Iterator<Item> I = new Catalog().getAllItems().iterator() ; I.hasNext() ; ) {
                Item item = I.next();
            %>
            <tr><td><%= item.getName() %></td><td><%= item.getDescription() %></td><td><
            %>
            <%= item.getFormattedHeat() %></td><td><%= item.getFormattedPrice() %>
            </td><td><button onclick="addToDish('<%= item.getCode() %>')">来一份
            </button></td></tr>
            <% } %>
        </tbody>
    </table>

```

```

</table>
<button onclick="emptyDish()">重新点餐</button>
<div style="position: absolute; top: 0px; right: 0px; width: 250px">
<h2>已点菜目</h2>
<ul id="contents">
</ul>
餐费合计: <span id="total">0.00 元</span><br>
热量合计: <span id="heat">0.00 卡</span>
</div>
</body>
</html>

```

程序说明: 调用 JavaScript 脚本, 利用 Ajax 技术实现智能订餐功能。

运行结果

应用部署成功后, 在浏览器的地址栏输入“http://localhost:8080/Ajax/”, 将得到如图 1-29 所示的运行结果。单击相应菜品后面按钮【来一份】, 可以看到屏幕右侧的【已点菜目】列表在不断刷新。



图 1-29 利用 Ajax 实现智能点餐界面

讨论与思考

首先总结一下 Ajax 开发基本思路。

从上面的示例程序可以看出, Ajax 程序开发的基本思想是: 利用 JavaScript 向服务器端应用程序发出请求, 服务器端处理 JavaScript 请求的方法与处理一般 HTTP 请求的方法没什么不同, 只是它只能以 XML 文件的形式向脚本返回响应。JavaScript 使用指定的方法处理响应 XML 文件, 并根据其内容对当前页面进行刷新。由于页面实现采用 DHTML, 可以在不中断用户操作的情况下对页面局部进行刷新, 从而给用户带来卓越的体验感受。

Ajax 应用与传统应用程序的区别主要表现为: 一个传统 Web 应用程序模型实际上是一种基本的事件模型——请求响应模型, 即用户被迫提交请求以实现页面更新。这是与 Ajax 截然不同的, 在 Ajax 模

式下,数据交换经由特定的浏览器对象 XMLHttpRequest 实现,再由适当的逻辑来处理每个数据请求的结果,页面的特定区域而不是完整的页面被更新。结果是更快的速度、更少的拥挤和更好的信息传送控制。

传统请求响应模型的 Web 应用程序强迫用户中断工作过程而等待页面的重装。通过引入 Ajax 技术,一个客户端脚本能够异步地与服务器通话,而用户仍能保持输入数据。除了对用户透明之外,这样的异步意味着服务器可以有更多时间来处理请求。

传统的 Web 应用程序把所有的处理代理到服务器并且强迫服务器进行状态管理。Ajax 允许灵活划分应用程序逻辑以及客户和服务端之间的状态管理。这就消除了对请求响应模式的依赖性并且能够提供更好的服务器可伸缩性。当该状态存储在客户端时,开发人员就不必跨越服务器来维护应用程序的状态信息。

但任何技术都不是完美的, Ajax 技术也明显存在以下缺点:

(1) 对程序架构设计的破坏

由于使用了 Ajax,表现层的脚本编写量增加了, JSP 要求大量的 JavaScript 代码,应用程序的设计重新回到了单一 JSP (一堆 HTML、CSS 代码、图片和脚本代码),这大大破坏了 Java EE 分层的设计思想。

(2) 客户端影响

在向 Web 应用程序引入 Ajax 时,开发人员需要注意以下的风险:

- ① 可能没打开脚本功能:出于各种原因,在许多用户的浏览器上禁止了 JavaScript 支持。
- ② 跨浏览器支持增加了代码需求:支持多种浏览器和多个浏览器版本的应用程序,要求的脚本代码可能会增多,因为浏览器解释 DOM 对象的方式有细微的差异(所以操作这些元素的 JavaScript 代码也有差异)。

③ JavaScript 不安全:在多数浏览器中,可以选择查看源代码选项,查看到与 HTML 页面关联的 JavaScript 源代码。在使用 Ajax 模式时,要确保脚本代码中实现的逻辑不是敏感逻辑。

(3) 线程问题

在典型的同步 Web 应用程序中,有些领域对按钮或链接单击要求更长一点的处理时间。没有耐心和没经验的 Web 用户通常会不止一次地单击按钮或链接,以为可以帮助加快处理速度,从而引发多重表单提交。Web 应用程序中的多重表单提交在某些情况下是无害的。而在其他情况下,则可能造成严重的线程问题或争用情况(此时多个线程竞争执行一个代码块)。

在 Ajax Web 应用中,客户端与服务器之间既支持同步通信模型又支持异步通信模型, Ajax 应用程序在某个页面上可能混合了服务器端调用(即,或者同步,或者异步,或者混合了同步和异步)。如果它的功能没有正确分析和规划,那么也可能造成严重的线程问题或争用,异步调用可能处理得慢些。如果应用程序不做预防,用户可能会在异步线程正在处理时又调用了同步调用,因为页面没有刷新,所以无法阻止页面上的进一步活动。结果是两个线程并发处理,造成业务逻辑的混乱。

(4) 性能缺陷

使用 Ajax 还有可能影响基于 Java EE Web 的应用程序的性能。

首先,Servlet 容器的线程池可能受到影响。线程池指定 Web 容器中允许并发运行的线程的最大数量。每个客户机请求需要一个线程。但是,一个客户机请求不一定等于一个用户请求。浏览器可能为一个用户请求要求多个客户机请求。例如,提交表单的一个用户可能要求多个客户机请求(其中包含提交表单的值、检索 GIF 文件、检索 JavaScript 文件、检索 CSS 文件)。如果允许并发地提交同步和异步请求,就意味着每个用户请求至少要支持多出一个的线程消耗(用于 Ajax 请求)。虽然为每个用户请求多增加一个线程的可能性看起来不多,但是当应用程序处在负载之下时,影响就明显了(这时每个用户请求多出的一个额外线程乘上平均用户数量)。显然,这种情况有可能影响 Servlet 容器的性能。

另一个可能受影响的资源是数据库连接池。典型的 Java EE Web 应用程序支持一个用户请求的两类序列:浅(shallow)请求和深(deep)请求。浅请求是执行服务器端代码但是不访问持久性存储(如

数据库)就完成请求的 Web 页面发出的请求。深请求是执行服务器端代码并访问持久性存储才能完成请求的 Web 页面发出的请求。

在深请求序列中(假设需要数据库连接),数据库连接池的以下方面可能会由于允许多个线程而受到影响:

- ✎ 等待连接的线程的平均数量;
- ✎ 以毫秒为单位的连接的平均等候时间;
- ✎ 连接被使用的平均时间。

所以,可能需要提高连接池的平均大小或连接数量。

知识点索引

Ajax; 线程; 泛型。

本章小结

J2EE Web 应用体系架构通常分为表现层、控制层、业务逻辑层和持久化存储层。俗话说“人靠衣裳马靠鞍”,表现层是直接与客户进行交互的场所,关系到用户的切身体验和应用程序被接受的程度,因此表现层的实现也就成为决定 Web 应用系统成败的重要因素。

本章通过 12 个示例演示了表现层的编程方法和技巧。在 Java EE 体系架构下,表现层的实现技术多种多样,总结起来,有以下几种技术路线:基于 JSP、Servlet、JSTL、自定义标记库等经典 Java EE 技术动态生成 HTML 页面;基于 XML、Xalan、XSL、CSS 等 XML 技术生成动态页面;基于 Flash 技术生成动态页面。其中第一种技术是最成熟,也是应用最广泛的一种技术,后两种技术都是新兴的表现层实现技术,在开发工具支持、性能效率上相对前者还有一定差距。

为了使前端的表现形式更为丰富,本章通过示例还演示了如何使用 JavaScript、Applet、Java 图像、自定义标记库、Ajax 等技术来丰富表现层实现。

第2章 服务器与客户端的交互

知己知彼，百战不殆

——《孙子兵法·谋攻》

Web 应用程序的工作过程就是服务器与客户端浏览器往复交互过程，因此利用 Java EE 技术实现对服务器与客户端之间的交互控制便成为 Java EE 编程的核心任务。服务器与浏览器交互的基础便是互相了解对方的信息，本章将通过具体的示例演示 Java EE 体系中如何控制和管理服务器与浏览器之间信息交互的基本方法和技巧。

例程 2-1：奥运网上问卷调查

目的

- (1) 演示如何通过表单向服务器提交信息；
- (2) 演示服务器端处理客户端提交的信息；
- (3) 利用会话信息防止表单重复提交。

问题

如何实现一个关于奥运话题的网上调查问卷，并通过它来收集网上客户提交的反馈信息？

解决方案

前端利用 HTML 页面中的表单来显示调查问卷的内容，后端利用 Servlet 组件来处理客户提交的反馈信息。

实现步骤

- (1) 创建问卷信息显示页面 `vote.jsp`；
- (2) 创建处理问卷提交信息的 Servlet 组件 `FormProcess`。

示例代码

本例程的所有示例代码均在 netbeans 的工程 Servlet 内。

程序 2-1：`vote.jsp`

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
```

```

"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>关于北京奥运会的问卷调查</title>
  </head>
  <body>
    <form name="form1" method="post" action="FormProcess">
      姓名 <input type="text" name="name" value="" />
      性别 <select name="sex">
        <option value="男">男</option>
        <option value="女">女</option>
      </select>
      <br>
      <font color="#007ece"> 1 关于奥运期间是否放假的问题, 你认为<br></font>
      <input type="radio" name="vacation" value="全部放假, 可以好好观看比赛" />全部放假, 可以好好观看比赛<br>
      <input type="radio" name="vacation" value="放什么假啊, 工作才是第一" />放什么假啊, 工作才是第一<br>
      <input type="radio" name="vacation" value="除餐饮、服务、公安等特殊行业外, 其他都放假" />除餐饮、服务、公安等特殊行业外, 其他都放假<br>
      <input type="radio" name="vacation" value="放不放假关我啥事" />放不放假关我啥事<br>
      <font color="#007ece"> 2 北京奥运会你最希望得到的门票<br></font>
      <input type="checkbox" name="checkbox1" value="开幕式">
        开幕式
      <input type="checkbox" name="checkbox1" value="闭幕式">
        闭幕式
      <input type="checkbox" name="checkbox1" value="体操">
        体操
      <input type="checkbox" name="checkbox1" value="游泳">
        游泳
      <input type="checkbox" name="checkbox1" value="乒乓球">
        乒乓球
      <input type="checkbox" name="checkbox1" value="足球">
        足球
      <input type="checkbox" name="checkbox1" value="篮球">
        篮球
      <input type="checkbox" name="checkbox1" value="排球">
        排球
      <br>
      <input type="submit" name="Submit" value="提交">
      <input type="reset" name="reset" value="重置">
    </form>

  </body>
</html>

```

程序说明: 页面向服务器提交信息的最根本方法就是通过 Form 表单。所有包含在标签<form></form>内的组件的值都会提交到后端的服务器组件。Form 表单的 action 属性的值即为处理表单提交数据的服务

器组件的 URL 地址。

程序 2-2: FormProcess.java

```
package example.servlet;
import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class FormProcess extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse
    response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        request.setCharacterEncoding("UTF-8");
        String SubmitFlag=(String)request.getSession().getAttribute("OlympicForm");
        if(SubmitFlag==null){
            request.getSession().setAttribute("OlympicForm","1");

            String name=request.getParameter("name");
            if(name==null)name="";
            String sex=request.getParameter("sex");
            if(sex==null)sex="男";

            if(sex.equals("男"))name+="先生";
            if(sex.equals("女"))name+="女士";
            String holiday=request.getParameter("vacation");
            if(holiday==null)holiday="";
            String[] paramValues = request.getParameterValues("checkboxx1");
            String tickets=new String("");
            if(paramValues!=null)
            for(int i=0;i<paramValues.length;i++)tickets+=paramValues[i]+" ";
            PrintWriter out = response.getWriter();

            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet FormProcess</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>欢迎您! " +name+ "</h1>");
            out.println("<h3>1 您对关于奥运期间是否放假的问题的看法是<br> " +holiday+ "</h3>");
            out.println("<h3>2 北京奥运会你最希望得到的门票是 <br> " +tickets+ "</h3>");
            out.println("</body>");
            out.println("</html>");

            out.close();
        }
        else{
            PrintWriter out = response.getWriter();
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet FormProcess</title>");
            out.println("</head>");
```

```

        out.println("<body>");
        out.println("<h1>对不起! 您的信息已经提交过了, 谢谢参与! </h1>");
        out.println("</body>");
        out.println("</html>");

        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    public String getServletInfo() {
        return "Short description";
    }
}

```

程序说明: 程序实现了对客户端提交数据的提取和处理。为了正确地提取信息, 防止出现乱码, 首先调用 `HttpServletRequest` 对象的 `setCharacterEncoding("UTF-8")` 方法来设置请求的编码格式, 然后调用 `HttpServletRequest` 对象的 `getParameter()` 方法获取文本框、下拉列表框、单选按钮的值; 调用 `HttpServletRequest` 对象的 `getParameterValues()` 方法获取多选按钮的值。最后利用 `HttpServletResponse` 对象的 `PrintWriter` 将信息显示到页面。为正确显示文字信息, 必须调用 `HttpServletResponse` 对象的 `setContentType("text/html; charset=UTF-8")` 方法设置输出内容的类型和编码格式。

为了防止由于客户端连续单击提交按钮或者刷新页面引起的重复提交, 程序在客户的会话信息中设置了一个标志变量 `OlympicForm`。在处理客户端提交的信息之前, 首先检查此变量, 如果变量存在, 则表明客户已经提交过表单信息, 则将错误提示信息返回到客户端浏览器即可; 否则, 将标志变量添加到会话中, 并着手处理客户端提交的信息。

运行结果

程序发布成功后, 在浏览器地址栏输入 “`http://localhost:8080/Servlet/vote.jsp`”, 将得到如图 2-1 所示的运行结果。

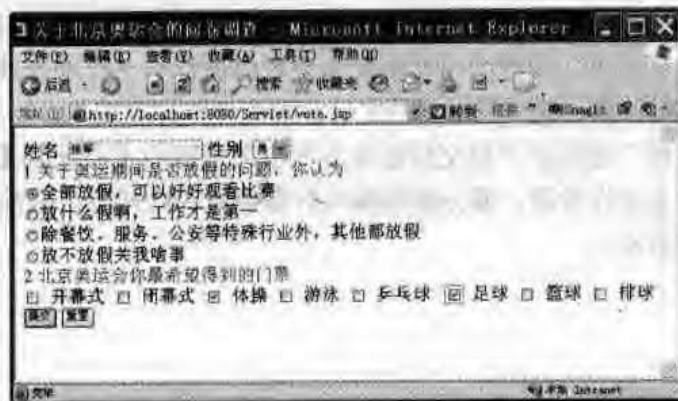


图 2-1 调查问卷页面

完成调查问卷,单击【提交】按钮,问卷信息提交到 Servlet FormProcess 处理,最后得到如图 2-2 所示的显示页面。可以看到提交的信息已经成功获取。

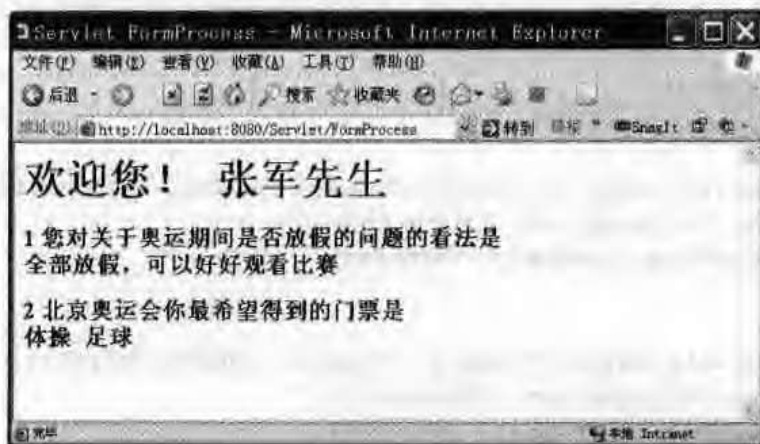


图 2-2 显示获取的问卷信息

单击浏览器的【后退】按钮回到图 2-1 所示的页面。重新单击【提交】按钮来提交信息,将得到如图 2-3 所示运行结果。



图 2-3 对重复提交显示的错误提示信息

讨论与思考

通过表单向服务器端提交信息是客户端与服务器进行交互的最基本方式。

服务器端只要简单地调用 `HttpServletRequest` 对象的 `getParameter()` 方法或 `getParameterValues()` 方法,给出参数名称即可取得该参数变量的值。需要注意的是,参数的名称是大小写敏感的。当请求的变量不存在时,将会返回一个空字符串,因此必须对获取的参数值进行空字符串处理。另外,对于提交的变量信息,必须采用正确的编码处理,这一点对于中文信息的处理特别重要。

另外对于表单,由于用户连续单击提交按钮或者单击浏览器的【前进】、【后退】等功能按钮,可能造成信息的重复提交,为安全性考虑,服务器端编码时需要解决此问题。最简单的方法就是在会话中设置一标记变量,如本例中所示。

知识点索引

表单处理: JSP; Servlet。

例程 2-2：发送 PDF 文件到客户端浏览器

目的

- (1) 演示如何获取 Servlet 配置参数信息；
- (2) 演示如何发送非 HTML 文件到客户端浏览器。

问题

Web 应用程序中，如何利用 Servlet 技术把位于特定路径下的非 HTML 文件（如图像文件、Word 文档、PDF 文档等）传递到浏览器客户端？

解决方案

首先获取要发送的文件的完整路径名称，根据此名称创建一个文件输入流对象，然后根据 `HttpServletResponse` 对象创建一个到客户端浏览器的输出流对象，最后，将文件输入流对象中的数据发送到输出流对象即可。为了保持应用的可移植性，代码中避免将文件路径信息以硬编码的方式编写在代码中，而是以 Servlet 参数的形式提供。

知识链接

Servlet 配置参数是包含在 Web 应用配置文件 `Web.xml` 中供 Servlet 初始化调用的信息。Servlet 除了从请求对象中获取信息以外，还可以从配置文件中获取配置参数信息。与请求中的动态信息不同，配置文件中的参数信息与具体的请求无关，而是 Servlet 初始化时调用的。Servlet 每次启动都会重新加载初始化参数。

实现步骤

- (1) 创建 Servlet `ParameterServlet` 实现 PDF 文件的发送操作；
- (2) 在配置文件 `web.xml` 中为 Servlet 添加初始化参数 `Path`，代表要发送的文件路径信息。添加后的文件信息如程序 2-4 所示。

示例代码

本例程的所有示例代码均在 netbeans 的工程 `Servlet` 内。

程序 2-3: `ParameterServlet.java`

```
package example.servlet;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class ParameterServlet extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse res)
        throws ServletException, IOException {
```



```

res.setContentType( "application/pdf");
ServletOutputStream out =res.getOutputStream();
File pdf =null;
BufferedInputStream buf=null;
try{
    String path=getInitParameter("path");
    pdf=new File(path);//为演示 PDF 文件发送而保存的一个文件
    res.setContentLength((int) pdf.length());
    FileInputStream input = new FileInputStream(pdf);
    buf = new BufferedInputStream(input);
    int readBytes = 0;
    while((readBytes = buf.read( )) != -1)
        out.write(readBytes);
} catch(IOException e){
    System.out.println("file not found!");
}finally {
    if (out != null)
        out.close( );
    if (buf != null)
        buf.close( );
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}
}

```

程序说明：代码主要实现发送 PDF 文件到客户端浏览器的功能。为了确保客户端浏览器能够正确处理发送来的 PDF 文件，首先调用 `HttpServletResponse` 对象的 `setContentType("application/pdf")` 方法将响应内容类型定义为 PDF 类型，然后调用 `getOutputStream()` 方法获取 `ServletOutputStream`。为使 PDF 文件以流的形式输出到客户端，调用 `getInitParameter()` 方法从 Servlet 配置参数获取文件路径名称，并以此作为参数生成一个 `File` 对象，根据 `File` 对象得到一个 `FileInputStream` 对象。为确保发送正确，获取 `File` 对象的长度信息，并调用 `HttpServletResponse` 对象的 `setContentLength()` 方法对服务器响应的长度进行设置。通过将 `FileInputStream` 中的信息写到 `ServletOutputStream` 中实现 PDF 文件的发送。`BufferedInputStream` 对象只是起到缓冲的作用。

程序 2-4: webxml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http:

```

```

//java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <servlet>
    <servlet-name>ParameterServlet</servlet-name>
    <servlet-class>example.servlet.ParameterServlet</servlet-class>
    <init-param>
      <param-name>path</param-name>
      <param-value>d:\\temp\\sample.pdf</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>ParameterServlet</servlet-name>
    <url-pattern>/ParameterServlet</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>
      index.jsp
    </welcome-file>
  </welcome-file-list>
</web-app>

```

程序说明：斜体加粗部分为添加的 Servlet 初始化参数。注意：Java 字符串中必须使用“\\”来代表文件路径分割符号。

运行准备

在目录 d:\temp\下放置一个文件 sample.pdf。

运行结果

程序发布成功后，在浏览器地址栏输入“http://localhost:8080/Servlet/ParameterServlet”，将得到如图 2-4 所示的运行结果，可以看到 PDF 文件已经成功地发布到客户端浏览器。

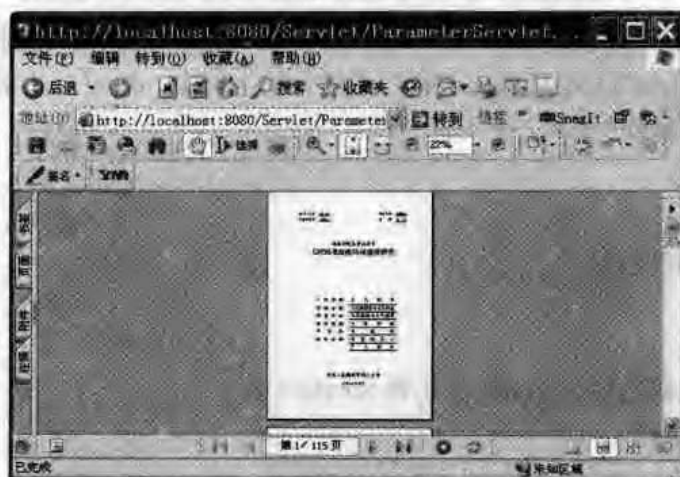


图 2-4 发布 PDF 文件到客户端浏览器

讨论与思考

对于非 HTML 文件到浏览器客户端的发送,由于文件的体积一般较大,通常采用流的方式进行传输。发送文件创建一个输入流,服务器的响应对象创建一个输出流,这样就在服务器与浏览器之间建立起一个流通道。

Servlet 已经实现了 ServletConfig 接口,因此可以在任何地方调用 `getInitParameter()` 方法获得 Servlet 初始化参数。通过配置参数来初始化 Servlet 可以有效避免硬编码信息,提高 Servlet 的可移植性。Servlet 配置参数由 ServletConfig 对象表示。在 Servlet 被实例化后,ServletConfig 对象对任何客户端在任何时候访问都有效,但一个 Servlet 的 ServletConfig 对象不能被其他 Servlet 访问。

使用配置参数来初始化 Servlet 的另一个特性是只有 Servlet 在加载时才会调用配置参数进行初始化,因此,配置参数仅适合表示一些固定的信息,而不是动态信息。

知识点索引

发送非 HTML 文档: Servlet; ServletConfig。

例程 2-3: 客户信息显示栏

目的

- (1) 掌握 Web 开发中获取客户端浏览器和操作系统等相关信息的方法;
- (2) Java 中字符串的操作。

问题

如何实现一个客户欢迎信息栏,在信息栏中显示客户的 IP 地址、操作系统名称、浏览器名称等基本信息?

解决方案

利用 `HttpServletRequest` 对象的 `getRemoteAddr()` 方法获取客户端的 IP 地址,利用 `HttpServletRequest` 对象的 `getHeader("user-agent")` 方法获得代表客户端信息的字符串,然后对此字符串进行分析,获取客户端的浏览器和操作系统信息。

实现步骤

创建 JSP 页面 `clientInfo.jsp`。

示例代码

本例程的所有示例代码均在 netbeans 的工程 Servlet 内。

程序 2-5: `clientInfo.jsp`

```
<%@page contentType="text/html"%>
```

```

<%@page pageEncoding="UTF-8"%>
<%@page import="java.util.StringTokenizer"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>显示客户端信息</title>
  </head>
  <body>
    <%
      String agent = request.getHeader("user-agent");
      StringTokenizer st = new StringTokenizer(agent, ";");
      st.nextToken();
      //得到用户的浏览器名
      String userbrowser = st.nextToken();
      //得到用户的操作系统名
      String useros = st.nextToken();
      String IP=request.getRemoteAddr();
    %>
    欢迎来自<%=IP%>的朋友，您使用的浏览器是<%=userbrowser%>，您使用的操作系统是<%=useros%>
  </body>
</html>

```

运行结果

程序发布成功后，在浏览器地址栏输入“<http://localhost:8080/Servlet/clientInfo.jsp>”，将得到如图 2-5 所示的运行结果，可以看到客户端的 IP 地址、浏览器版本和操作系统版本。



图 2-5 显示客户端机器信息

讨论与思考

利用 `HttpServletRequest` 对象的 `getRemoteAddr()` 方法获得的 IP 是不准确的，因为如果采用代理方式上网的话，获取的将是代理机器的 IP 地址。利用 `HttpServletRequest` 对象的 `getHeader("user-agent")` 方法获得客户端的浏览器和操作系统信息也不是浏览器和操作系统的准确名称，不过可以将它们映射到浏览器和操作系统的准确名称。

知识点索引

获取客户端信息；字符串操作。

例程 2-4：获取服务器基本信息

目的

演示如何获取 Servlet 运行所在的服务器的名称、端口等环境信息

问题

在 Java EE 组件中如何获取运行服务器的相关信息？

解决方案

调用 `HttpServletRequest` 对象的方法，`getServerName()`，`getServerPort()` 可以获取服务器的相关信息。

实现步骤

- (1) 创建 Servlet `BrowserServer`；
- (2) 调用 `HttpServletRequest` 对象的 `getServerName()` 方法获取服务器名称；
- (3) 调用 `HttpServletRequest` 对象的 `getServerPort()` 方法获取服务器端口；
- (4) 首先调用 `getServletContext()` 方法获取 `ServletContext` 对象，然后调用 `ServletContext` 对象的 `getServerInfo()` 方法获取服务器运行环境信息包括服务器软件名称、版本信息等；
- (5) 利用 `HttpServletResponse` 对象的 `PrintWriter` 将信息显示到页面。

示例代码

本例程的所有示例代码均在 netbeans 的工程 `Servlet` 内。

程序 2-6: `BrowserServer.java`

```
package example.servlet;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 *
 * @author hyl
 * @version
 */
public class BrowserServer extends HttpServlet {
    /** Processes requests for both HTTP GET and POST methods.
```

```

    * @param request servlet request
    * @param response servlet response
    */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        ServletContext context = getServletContext();

        out.println("<html>");
        out.println("<head>");
        out.println("<title>服务器信息</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h3>服务器名称: " + request.getServerName() + "</h3>");
        out.println("<h3>服务器端口: " + request.getServerPort() + "</h3>");
        out.println("<h3>服务器信息: " + context.getServerInfo() + "</h3>");
        out.println("</body>");
        out.println("</html>");

        out.close();
    }

    /** Handles the HTTP <code>GET</code> method.
     * @param request servlet request
     * @param response servlet response
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    /** Handles the HTTP <code>POST</code> method.
     * @param request servlet request
     * @param response servlet response
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    /** Returns a short description of the servlet.
     */
    public String getServletInfo() {
        return "Short description";
    }
}

```


运行结果

程序发布成功后,在浏览器地址栏输入“`http://localhost:8080/Servlet/ParameterServlet`”,将得到如图 2-6 所示的运行结果,可以看到 Java EE 服务器的基本信息已经输出到页面上。



图 2-6 显示获取的服务器信息

讨论与思考

Java EE 追求的一贯目标就是实现组件的最大移植性。它使得符合 Java EE 规范的 Web 应用可以不需要任何改动就可以运行在不同类型的 Java EE 服务器上。但这并不代表 Web 应用组件不需要了解它所运行的服务器的相关信息。例如,开发人员为了保护自己开发的组件版权,限制组件只能运行在某一特定类型的服务器上。而且在很多实际情况下,真正实现不作任何改动就可以将应用部署在任何 Java 应用服务器几乎是不可能的。

知识点索引

获取服务器信息。

例程 2-5: 横幅广告系统

目的

- (1) 客户端“拉”技术;
- (2) 利用 JSP、Servlet 显示动态图片信息;
- (3) 利用 JDBC 操作数据库信息。

问题

对于一个商务网站来说,广告系统是必不可少的。一个好的广告系统是一个网站稳定收入的基础。而横幅广告(banner)是网站广告中最常见的形式。一个好的广告系统应该满足以下条件。

(1) 部署灵活。网站的广告内容和客户是经常变动的。广告系统必须在不需要更改任何程序代码的情况下就可以进行部署调整。

(2) 访问记录。客户必须可以查阅广告信息的访问量。因此,要求对广告的单击记录必须准确实

时的记录下来。

(3) 动态刷新。网站的广告空间位置很宝贵。横幅广告必须做到动态刷新,即一个广告位置必须动态展示多条广告信息。

如何实现满足以上功能的横幅广告系统?

解决方案

横幅广告系统的核心功能是实现动态信息的显示以及客户点击信息的记录。Java EE 技术提供了 JSP 和 Servlet 来展示动态信息, JDBC 技术来连接存储持久信息的数据库系统。因此,例程将主要采用上述三种技术来实现一个横幅广告系统。

实现步骤

(1) 创建辅助工具组件 DBConnection 获取到后台数据库的连接。

(2) 创建 Servlet 组件 BannerServlet, 利用 DBConnection 从后台数据库中的广告信息配置表中随机抽取一则广告配置信息, 添加到请求属性中发送到前端页面 index.jsp 进行显示。

(3) index.jsp 根据 request 的属性信息动态显示广告图片, 并动态生成广告图片的链接地址和 JavaScript 脚本方法; 同时通过设置 head 属性实现到 BannerServlet 的自动转向。

(4) 当客户单击广告时, 首先执行 JavaScript 脚本方法, 在弹出的新窗口中显示广告对应的链接地址, 同时发送请求信息到 BannerServlet。

(5) BannerServlet 根据请求参数来判断是页面的自动转向请求还是客户单击发出的请求。如果是客户单击请求, 则记录客户的单击信息, 然后刷新广告栏显示内容, 否则, 直接刷新广告栏显示内容。

示例代码

本例程的所有示例代码均在 netbeans 的工程 Banner 内。

程序 2-7: DBConnection.java

```
package com.sample;
import java.sql.*;
public class DBConnection {

    /** Creates a new instance of DBConnection */
    public DBConnection() {
    }

    public Connection getConnection() { // 获取数据库链接
        java.sql.Connection conn;
        try {
            Class.forName("org.gjt.mm.mysql.Driver").newInstance();
            conn= java.sql.DriverManager.getConnection("jdbc:mysql://localhost/
            ad","root","root");
        } catch (java.sql.SQLException e) {
            System.err.println(e.toString());
            return null;
        } catch (Exception ea) {
```



```

        System.err.println(ea.toString());
        return null;
    }
    return conn;
}
}

```

程序说明：本程序是一个用来获取到数据库的连接的工具类。获取数据库连接的方法有多种，在后面的示例中将会介绍更优秀的数据库连接方案。程序采用最简单的方法，直接根据 JDBC URL 来获得数据库的连接。

创建数据库连接，分为以下几步：首先注册数据库驱动程序，然后根据具体的 JDBC URL 地址，调用 DriverManager 对象的 getConnection() 来获取一个代表数据库连接的 java.sql.connection 对象。

getConnection() 方法的第一个参数为 MySQL 数据库 AD 的 JDBC URL 地址，（如果使用的是 Oracle 数据库，那么 JDBC URL 是：“jdbc:oracle:thin@机器名:端口名:数据库名”）。第二个参数和第三个参数分别是数据库的用户名和密码，要修改为用户数据库的实际用户名和密码。

程序 2-8: ADBean.java

```

package com.sample;
//代表一幅广告
public class ADBean {
    private String url="";
    private String imgName="";
    private int adid=-1;

    /** Creates a new instance of ADBean */
    public ADBean() {
    }

    public ADBean(int id) { //获取制定 ID 的广告对象
        adid=id;
        populate(id);
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public String getImgName() {
        return imgName;
    }

    public void setImgName(String imgName) {
        this.imgName = imgName;
    }

    public int getAdid() {

```

```

        return adid;
    }

    public void setAdid(int adid) {
        this.adid = adid;
    }

    public void populate(int id){//根据数据库的内容随机初始化一个广告
        DBConnection db=new DBConnection();
        java.sql.PreparedStatement preparedStmt; //语句对象
        java.sql.ResultSet sqlRst; //结果集对象
        try{
            preparedStmt =db.getConnection().prepareStatement("select * from adinfo
            where adid = ? ");
            preparedStmt.setInt(1,id);
            //执行SQL 语句
            sqlRst=preparedStmt.executeQuery();
            while (sqlRst.next()) { //取得下一条记录
                //String name=new String(sqlRst.getString("firstname").getBytes("iso-8859-1"));
                url=sqlRst.getString("url");
                imgName=sqlRst.getString("ImgName");
                adid=sqlRst.getInt("adid");
            }
        }
        catch(Exception e){
            System.out.print(e.toString());
        }
    }
}

```

程序说明：本程序是一个实体 Bean，它代表一幅广告实体对象。它有三个属性 adid, url 和 imgName，分别代表广告的编号、链接地址和图片名称。Bean 提供了操作这些属性的 setter 和 getter 方法。最值得注意的是 populate() 方法，它的作用是利用数据库中的信息来填充一个广告实体 Bean。在 populate() 方法中，根据传递来的参数 id，进行数据库查询操作，并根据查询结果填充实体 Bean。

程序 2-9: BannerServlet.java

```

package com.sample;

import java.io.*;
import java.util.ArrayList;
import java.util.Date;
import java.util.Random;

import javax.servlet.*;
import javax.servlet.http.*;

public class BannerServlet extends HttpServlet {
    ServletConfig myconfig;
    private ArrayList ids;//存储所有广告的 ID
    /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response

```



```

*/
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

    String action=request.getParameter("style");

    if(action!=null&&action.equals("link")){//客户单击请求
        UpdateClickRecord(request);
    }
    refreshAD( request,response); //刷新广告显示

    // response.setContentType("text/html;charset=UTF-8");
    // PrintWriter out = response.getWriter();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public void init(ServletConfig config) throws ServletException {
    super.init(config);
    myconfig=config;
    ids=new ArrayList();
}

public void refreshAD(HttpServletRequest request, HttpServletResponse
response) {//刷新广告显示
    String imgpath=(String)request.getSession().getAttribute("IMG_PATH");
    if(imgpath==null){
        String ADDir = myconfig.getInitParameter("ADImgPath");
        request.getSession().setAttribute("IMG_PATH",ADDir);
    }

    int i= getRandomADID();//随机获取广告ID
    ADBean a=new ADBean(i);

    // HttpSession session=request.getSession();
    request.setAttribute("url",a.getUrl());
    request.setAttribute("img",a.getImgName());
    request.setAttribute("ID",Integer.toString(i));
    try{
        request.getRequestDispatcher("/index.jsp").forward(request,response);
    }catch(Exception e){
        System.out.println(e.toString());
    }
}

```

```

    }
    /** Returns a short description of the servlet.
    */
    public String getServletInfo() {
        return "Short description";
    }

    public int getRandomADID() { //获取随机广告ID
        if(ids.isEmpty()){
            DBConnection db=new DBConnection();
            java.sql.Statement stmt; //语句对象
            java.sql.ResultSet sqlRst; //结果集对象
            try{
                stmt =db.getConnection().createStatement(java.sql.ResultSet.TYPE_SCROLL_INSENSITIVE, java.sql.ResultSet.CONCUR_READ_ONLY);

                //执行SQL语句
                String sqlQuery="select adid from adinfo";
                sqlRst=stmt.executeQuery(sqlQuery);

                while (sqlRst.next()) { //取得下一条记录
                    ids.add(sqlRst.getInt("adid"));
                }

            } catch (Exception e){
                e.toString();
            }
        }
        //随机挑选数字
        int max=ids.size()-1;
        Random random=new Random();
        int temp=random.nextInt(max);
        return temp+1;
    }

    private void UpdateClickRecord(HttpServletRequest request){
        String IP=request.getRemoteAddr();
        String currentID=(String)request.getParameter("ID");
        if(currentID==null)return;
        int cid=Integer.parseInt(currentID);
        //////////////////////////////////////
        DBConnection db=new DBConnection();
        java.sql.PreparedStatement preparedStmt; //语句对象

        try{
            preparedStmt =db.getConnection().prepareStatement("insert into adclickinfo (adid,clientip,clicktime) values(?,?,now()) ");
            preparedStmt.setInt(1,cid);
            preparedStmt.setString(2,IP);
        }
    }

```



```

        //执行 SQL 语句
        int result=preparedStmt.executeUpdate();

    }
    catch(Exception e){
        System.out.println( e.toString());
    }
    ///////////////////////////////////////////////////
}
// </editor-fold>
}

```

程序说明：本程序是一个 Servlet，用来实现广告系统的核心功能，即生成动态显示的广告图片链接地址和记录客户的单击信息。由于客户每次单击后，广告栏显示都会自动刷新，因此，将上述两个功能集成在一个 Servlet 中实现。

processRequest()方法实现了对 GET 和 POST 请求的处理。它根据请求中的参数 style 来判断目前的请求类型。如果是客户单击，则调用 UpdateClickRecord(request)方法来记录客户的单击信息；否则，调用 refreshAD(request,response)方法来刷新广告显示。在 refreshAD()方法中，通过代码 request.getRequestDispatcher("/index.jsp").forward(request,response)将请求导向 JSP 页面。

客户端的请求信息通过 request 对象的参数传递到 Servlet，Servlet 的处理结果同样也通过 request 对象的参数传递到 JSP 页面。

在 UpdateClickRecord(request)方法中，由于要将单击时间插入数据库，例程中在 SQL 语句中直接使用了 MySQL 的内置函数 now()。这样既减少了代码量，又提高了 SQL 操作的执行效率。其他类型的数据库（如 Oracle）也提供了一套自己的内置函数。因此使用内置函数在提高效率的同时，也存在降低代码可移植性的缺点。

程序 2-10: StatServlet.java

```

package com.sample;
import java.io.*;
import java.util.ArrayList;

import javax.servlet.*;
import javax.servlet.http.*;

public class StatServlet extends HttpServlet {

    protected void processRequest (HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        ArrayList ip=new ArrayList();
        ArrayList time=new ArrayList();
        String ID=request.getParameter("ID");
        if(ID==null){

            response.setContentType("text/html;charset=UTF-8");
            PrintWriter out = response.getWriter();
            out.println("<html>");
            out.println("<head>");
            out.println("<title>广告统计</title>");
            out.println("</head>");

```

```

        out.println("<body>");
        out.println("<h1>尚未提交要统计的广告 ID... 正确方法如下: </h1>");
        out.println("<h2>尚未提交要统计的广告 ID... </h2>");
        out.println("</body>");
        out.println("</html>");

        out.close();
    } else {
        DBConnection db=new DBConnection();
        java.sql.PreparedStatement preparedStmt; //语句对象
        java.sql.ResultSet sqlRst; //结果集对象
        try{
            preparedStmt =db.getConnection().prepareStatement("select clientIP,
            clicktime from adclickinfo where adid=? ");
            preparedStmt.setInt(1,Integer.parseInt(ID));
            sqlRst=preparedStmt.executeQuery();

            while (sqlRst.next()) { //取得下一条记录
                ip.add(sqlRst.getString("clientIP"));
                time.add(sqlRst.getDate("clicktime"));
                time.add(sqlRst.getTime("clicktime"));
            }

        } catch(Exception e){
            System.out.println( e.toString());
        }
    }
    request.setAttribute("IP",ip);
    request.setAttribute("time",time);
    try{
        request.getRequestDispatcher("/ADStat.jsp").forward(request,response);
    }catch(Exception e){
        System.out.println(e.toString());
    }
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}

```



```

}

```

程序说明：用来对指定 ID 的广告的单击记录实现查询统计，广告的 ID 通过请求参数来提供。

程序 2-11: index.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%
String imgpath=(String)request.getSession().getAttribute("IMG_PATH");//获取当前图像路径
String url=(String)request.getAttribute("url");
//url+="BannerServlet?style=link";
String img=(String)request.getAttribute("img");
String ID=(String)request.getAttribute("ID");
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="refresh" content="20;url=http://localhost:8080/Banner/
BannerServlet; charset=UTF-8">
<title>横幅广告</title>
</head>
<body>

<h3>欢迎来到本网站....</h3>
<%String s=request.getContextPath();
s+=imgpath+"/"+img;
String href="http://localhost:8080/Banner/BannerServlet?style=link&ID=";
href+=ID;
%>

<a href="<%=href%>"> <IMG ONCLICK="OpenAD()"height=50 src="<%=s%>" width=145> </a>

</body>
<script>
function OpenAD(){
window.open('<%=url%>',width=400,height=300);
}
</script>
</html>

```

程序说明：页面用来显示动态广告信息。页面首先从请求中获取广告的相关信息，然后动态生成广告对应的链接地址和图片地址。通过 URL 重写的方法在链接地址中包含了当前显示广告的 ID 信息。广告被单击时，将首先执行脚本方法，在弹出的页面中显示广告对应的页面地址，同时发送请求到后台的

BannerServlet。页面通过设置<meta http-equiv="refresh" content="20;url=http://localhost:8080/Banner/BannerServlet; charset=UTF-8">实现自动转向到后台的组件 BannerServlet，使得广告栏即使没有客户单击，也可实现自动刷新。

在脚本方法 OpenAD()中，还演示了在 JavaScript 中使用 JSP 表达式引用 Java 变量的方法。

运行准备

步骤 1: 建立后台数据库

打开 MySQL 数据库，首先建立一个横幅广告系统专用的数据库 AD。系统中共有两个表格：广告信息表 adinfo 用来存储广告的配置信息，广告单击记录信息表 adclickinfo 用来存储广告的单击记录。表格的详细信息分别如表 2-1 和表 2-2 所示。

表 2-1 广告信息配置表 adinfo 的结构信息

字段名	字段类型	备注	字段说明
Adid	Integer	主键，系统自增字段	广告信息的标志 ID
url	Varchar (45)		广告的链接地址
ImgName	Varchar (45)		广告图像的文件名（不含路径）

表 2-2 广告信息配置表 adclickinfo 的结构信息

字段名	字段类型	备注	字段说明
cid	Integer	主键，系统自增字段	单击记录标志 ID
Adid	Integer	外键，引用 adinfo 中的 Adid	广告信息的标志 ID
clientIP	Varchar (45)		单击客户的 IP 地址
clickTime	Datetime		客户单击时间

提示：可以根据以上信息在数据库中建立相应表格。也可通过脚本文件 ad.sql 来还原数据库。MySQL 有很多客户端软件（如 EngInSite MySQL Client）可以帮助开发人员还原数据库。

步骤 2: 准备广告展示图片

(1) 在实例工程的根目录下建立一个存放广告图片的目录 dir，并将相关的示例广告图片放入目录下。

(2) 将广告信息填充到广告配置信息表 adinfo 中。广告配置信息主要分为两部分：广告的图片名称和广告对应的链接地址。SQL 语句的形式如下形式：

```
insert into adinfo (url,imgname)values('http://www.163.com','1.gif');
```

步骤 3: 添加数据库驱动包到工程的编译路径

要成功访问后台数据库，必须将数据库驱动程序添加到工程的 classpath 路径下。具体操作步骤如下：在【项目】视图中，选中项目文件夹【Banner】下的【库】文件夹，单击右键，在弹出的快捷菜单中选中【添加 JAR/文件夹】选项，如图 2-7 所示。

通过文件浏览，选中 MySQL 的 JDBC 驱动程序，如图 2-8 所示。单击【打开】按钮，则将数据库的驱动程序包添加到了工程的编译路径中。打开【库】文件夹，在下面可以看到新增的 MySQL 的 JDBC 驱动程序。

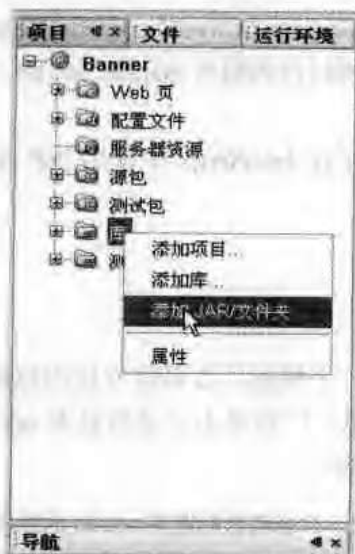


图 2-7 为工程添加 Jar 文件



图 2-8 添加数据库驱动包

运行结果

应用部署成功后，在浏览器的地址栏输入“<http://localhost:8080/Banner/BannerServlet>”，将得到如图 2-9 所示的运行结果。



图 2-9 程序运行结果画面

可以观察到广告栏在间隔一段时间就自动刷新。单击广告栏显示图片，则弹出一新窗口来显示广告链接的内容，如图 2-10 所示。



图 2-10 弹出的链接广告画面

在地址栏中输入“`http://localhost:8080/Banner/StatServlet?ID=2`”，则可得到编号为 2 的广告的单击记录信息，如图 2-11 所示。

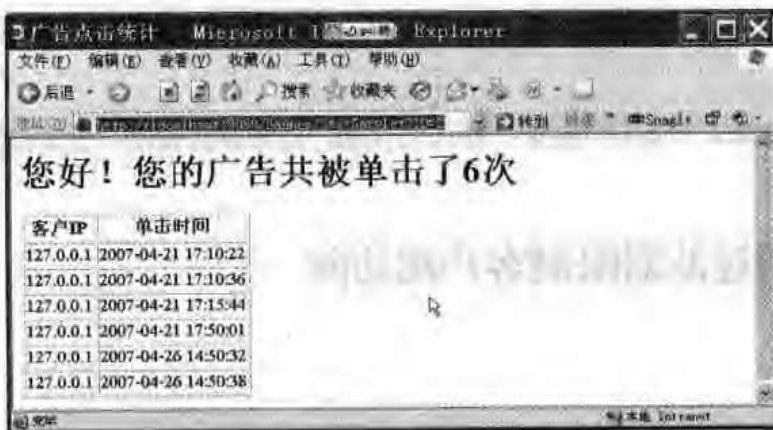


图 2-11 显示广告单击记录信息

讨论与思考

在上面的示例中，通过 JSP、Servlet 和 JDBC 技术实现了广告信息的动态显示。其中，JSP 页面主要实现广告信息的动态展示，Servlet 组件完成了绝大部分的业务逻辑，包括广告的随机选取、广告单击信息的记录、统计等。这是由这两种技术的应用特点决定的。因为 JSP 更便于信息的展示，而 Servlet 在业务处理控制方面的功能更强大。

JSP 实现动态页面的根本思想在于利用 JSP 脚本、表达式等技术生成动态 HTML 标记并输出到页面。

JSP 与 Servlet 之间的交互通过以下方式实现：JSP 页面通过自动转向向 Servlet 发出请求（更常见的方法是通过表单提交或 forward 指令向 Servlet 发出请求），Servlet 调用 request 的 `getRequestDispatcher()` 方法获取指定 JSP 页面的 `Http RequestDispatcher` 对象，然后调用 `Http RequestDispatcher` 对象的 `forward()` 方法将请求返回 JSP 页面。

为了实现广告显示页面的自动刷新,这里使用了 HTML 的<head>标记的属性。另外,还利用 JavaScript 脚本在弹出窗口中显示广告链接内容。因此,要设计一个强大的动态 Web 网站,不仅需要掌握服务器端开发技术,同时还要掌握浏览器端的网页开发技术。

最后,还需要对例程中持久化存储的设计做一下说明。几乎任何信息系统都存在持久化信息的存储。广告管理系统中需要持久化存储的信息共有两种:广告的配置属性信息和客户的单击记录信息。持久化信息可以存放在属性文件中,也可以存储在数据库中。如果采用属性文件来存储持久信息,由于客户单击记录在广告系统实际运行过程中需要不断刷新,频繁的输入/输出操作将影响系统的性能,同时多个客户的并发单击动作还需要对属性文件读写操作进行并发处理,因此,将系统中的持久化信息统一存放在后台数据库中(在本例中采用 MySQL 数据库),由数据库系统帮开发者完成并发处理等复杂的底层操作,这将大大提高系统运行效率,降低系统开发难度。

每则广告都有一个展示图片,但描述广告配置信息的数据库表格并不存储图片信息本身,而是仅仅保存图片的名称。广告的图片信息直接存放在网站的特定目录下,主要是为了提高广告图片的加载效率,减轻数据库的负荷,因为直接从文件系统加载广告图片要比频繁通过 SQL 操作检索,然后利用数据流输出 BLOB 图像对象的解决方案在性能上要优化得多。

Java EE 数据库的访问方式有多种,由于本例程的目的主要是演示如何利用 JSP、Servlet 展示动态信息,因此采用最直接的方式来操作数据库,并没有对数据库访问方式进行深入讨论。在随后的例程中还将对此问题进行深入探讨研究。

知识点索引

JSP; Servlet; JDBC; 页面自动刷新; 持久化存储; 请求参数处理; 会话管理; JSP 与 JavaScript 的交互。

例程 2-6: 利用过滤器限制客户端访问

目的

演示如何使用过滤器过滤客户端请求。

问题

假设一个网上资料库应用程序,设计人员仅仅希望它允许特定的用户访问,那么如何应用 Java EE 技术来实现这一功能?

解决方案

将允许访问的客户 IP 信息存放在列表中,从请求对象中获取客户访问请求的 IP 信息,然后利用过滤器,根据客户的 IP 信息是否在列表内决定是否允许访问资源。

知识链接

Java Servlet 规范 2.3 之后的版本增加了不少新特性,其中之一便是过滤器 (Servlet Filter)。Filter 可以改变一个请求 (Request) 或者是修改响应 (Response)。注意: Filter 不是 Servlet,它只是 Servlet 接收请求前的预处理器。Filter 与 Servlet 的关联由 Web 应用的配置描述文件来明确。用户发送请求给 Servlet

时,在Servlet处理请求之前,与此Servlet关联的Filter首先执行,然后才是Servlet的执行。如果一个Servlet有多个Filter,则在Web.xml文件中,靠前的Filter先执行。

实现步骤

- (1) 创建过滤器IPFilter;
- (2) 在Web.xml文件中配置过滤器信息。

示例代码

本例程的所有示例代码均在netbeans的工程IPFiltert内。

程序 2-12: IPFilter.java

```
package com.eaxmple;

import java.io.*;
import java.net.*;
import java.util.*;
import java.text.*;
import javax.servlet.*;
import javax.servlet.http.*;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class IPFilter implements Filter {

    private FilterConfig filterConfig = null;
    private ArrayList IPList=new ArrayList();
    public IPFilter() {
    }

    private void doBeforeProcessing(ServletRequest request, ServletResponse response)
    throws IOException, ServletException {
        if (debug) log("IPFilter:DoBeforeProcessing");
    }

    private void doAfterProcessing(ServletRequest request, ServletResponse response)
    throws IOException, ServletException {
        if (debug) log("IPFilter:DoAfterProcessing");
    }

    public void doFilter(ServletRequest request, ServletResponse response,
```



```

        FilterChain chain)
        throws IOException, ServletException {
            if (debug) log("IPFilter:doFilter()");

        doBeforeProcessing(request, response);
            Throwable problem = null;

        try {
            response.setContentType("text/html;charset=GBK");
            PrintWriter out = response.getWriter();
            String remoteAddr = ((HttpServletRequest) request).getRemoteAddr();
            System.out.println(remoteAddr);
            if (IPList.indexOf(remoteAddr)==-1) { //
                out.println("内部资料, 禁止访问!");
                out.close();
                return;
            } else{
                //out.println("通过认证!");
                chain.doFilter(request, response);
            }
        } catch (Exception e) {
            System.out.println(e.toString());
        }

        doAfterProcessing(request, response);

        if (problem != null) {
            if (problem instanceof ServletException) throw (ServletException)problem;
            if (problem instanceof IOException) throw (IOException)problem;
            sendProcessingError(problem, response);
        }
    }

    public FilterConfig getFilterConfig() {
        return (this.filterConfig);
    }

    public void setFilterConfig(FilterConfig filterConfig) {

        this.filterConfig = filterConfig;
    }

    public void destroy() {
    }

    public void init(FilterConfig filterConfig) {

        this.filterConfig = filterConfig;
        if (filterConfig != null) {
            if (debug) {
                log("IPFilter:Initializing filter");
            }
        }
    }

```

```

    }
    IPList.add("127.0.0.1");
}

/**
 * Return a String representation of this object.
 */
public String toString() {

    if (filterConfig == null) return ("IPFilter()");
    StringBuffer sb = new StringBuffer("IPFilter()");
    sb.append(filterConfig);
    sb.append(" ");
    return (sb.toString());

}

private void sendProcessingError(Throwable t, ServletResponse response) {

    String stackTrace = getStackTrace(t);

    if(stackTrace != null && !stackTrace.equals("")) {
        try {
            response.setContentType("text/html");
            PrintStream ps = new PrintStream(response.getOutputStream());
            PrintWriter pw = new PrintWriter(ps);
            pw.print("<html>\n<head>\n<title>Error</title>\n</head>\n<body>\n"); //NOI18N

            // PENDING! Localize this for next official release
            pw.print("<h1>The resource did not process correctly</h1>\n<pre>\n");
            pw.print(stackTrace);
            pw.print("</pre></body>\n</html>"); //NOI18N
            pw.close();
            ps.close();
            response.getOutputStream().close();
        }

        catch(Exception ex) { }
    } else {
        try {
            PrintStream ps = new PrintStream(response.getOutputStream());
            t.printStackTrace(ps);
            ps.close();
            response.getOutputStream().close();
        } catch(Exception ex) { }
    }
}

```



```

public static String getStackTrace(Throwable t) {

    String stackTrace = null;

    try {
        StringWriter sw = new StringWriter();
        PrintWriter pw = new PrintWriter(sw);
        t.printStackTrace(pw);
        pw.close();
        sw.close();
        stackTrace = sw.getBuffer().toString();
    } catch (Exception ex) {}
    return stackTrace;
}

public void log(String msg) {
    filterConfig.getServletContext().log(msg);
}

private static final boolean debug = true;
}

```

程序说明：程序包含了所有 Filter 所必须实现的三个接口方法：init()、destroy()和 doFilter()，其中编码主要集中在 doFilter()方法中。为实现过滤功能，在 doFilter()方法中首先调用 ServletRequest 对象的 getRemoteAddr()方法来获取请求的 IP 地址，然后根据 IP 地址判断是否为合法请求，如果不是，则返回显示错误信息；否则，调用 chain.doFilter(request,response)方法将请求传递到下一个 Filter 或 Servlet。

程序 2-13: Web.xml (片段)

```

<filter>
    <filter-name>IPFilter</filter-name>
    <filter-class>com.eaxmple.IPFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>IPFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>

```

程序说明：显示 Filter 的配置信息。可以看到对所有的请求，过滤器都将进行过滤。

运行结果

程序发布成功后，在浏览器地址栏输入“http://localhost:8080/IPFilter/”，将得到如图 2-12 所示的运行结果。

在局域网的另外一台机器上打开 IE 浏览器，在地址栏中输入“http://hyl:8080/IPFilter/”（在本例中，开发环境所在的机器名为“hyl”，试验时应该以实际的机器名或 IP 地址替换），得到如图 2-13 所示的运行结果。可以看到，由于 Filter 的作用，已经有效地屏蔽了远程地址发送的请求。



图 2-12 显示客户端机器信息



图 2-13 禁止远端用户访问

讨论与思考

Java Servlet 规范 2.3 之后的版本增加了不少新特性，其中之一便是 Servlet Filter，它其实就是管道和过滤器体系架构在 Java EE 中的应用实践。通过使用该模式使得 Web 应用开发者能够在请求到达 Web 资源之前截取请求，在处理请求之后修改应答。因此，它往往被应用到以下场景：

- ① 访问特定资源（Web 页、JSP 页、Servlet）时的身份认证；
- ② 应用程序级的访问资源的审核和记录；
- ③ 应用程序范围内对资源的加密访问，它建立在定制的加密方案基础上；
- ④ 对被访问资源的及时转换，包括从 Servlet 和 JSP 的动态输出。

知识点索引

过滤器。

例程 2-7：多组件协作实现用户登录验证

目的

演示服务器上的多个组件协作实现对客户端请求的响应。

问题

在网站常见的用户登录模块中，对客户端登录请求的响应往往是很复杂的，如果是合法用户，则导

向欢迎页面，如果是非法登录信息，则采取其他响应如重新导向登录页面等。如何实现上述功能的登录模块呢？

解决方案

除了用户登录信息提交页面外，创建三个不同的 Servlet 来实现用户登录验证功能，其中一个 Servlet 处理用户提交的登录信息并负责进行登录验证，另外两个 Servlet 分别代表登录成功和失败时对客户端的响应，三个 Servlet 通过请求指派对象 RequestDispatcher 联系起来。

实现步骤

- (1) 创建用户登录信息提交页面；
- (2) 创建处理用户登录请求的 Servlet 组件 Main；
- (3) 创建代表登录成功响应的 Servlet 组件 LoginSuccess；
- (4) 创建代表登录失败响应的 Servlet 组件 LoginFail。

示例代码

本例程的所有示例代码均在 netbeans 的工程 Servlet 内。

程序 2-14: login.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html lang='zh'>
  <head>
    <title>登录</title>
  </head>
  <body bgcolor="#FFFFFF">
    <center>欢迎登录系统</center>
    <form name="login" method="post" action="Main">
      <label>用户名: </label>
      <input type="text" name="userID" value="">
      <label>密 码: </label>
      <input type="password" name="password" value="">
      <input type="submit" name="tj" value="提交" ></input>
      <input type="reset" name="reset" ></input>
    </form>
  </body>
</html>
```

程序说明：用来提交用户登录信息。

程序 2-15: Main.java

```
package example.servlet;

import java.io.*;
import java.net.*;
```

```

import javax.servlet.*;
import javax.servlet.http.*;
public class Main extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        String userID=request.getParameter("userID");
        if(userID==null)userID="";
        String password=request.getParameter("password");
        if(password==null)password="";
        if((userID.equals("guest")&&password.equals("guest"))){
            RequestDispatcher dispatcher =
                request.getRequestDispatcher("LoginSuccess");
            dispatcher.forward(request, response);
        } else{
            RequestDispatcher dispatcher =
                request.getRequestDispatcher("LoginFail");
            dispatcher.forward(request, response);
        }
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    public String getServletInfo() {
        return "Short description";
    }
}

```

程序说明：用来处理客户端提交的登录请求，并根据登录信息的验证结果，调用 RequestDispatcher 对象的 forward() 方法，将请求导向其他 Servlet 组件。

程序 2-16: LoginSuccess.java

```

package example.servlet;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;
public class LoginSuccess extends HttpServlet {

```



```

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    String name=request.getParameter("userID");
    out.println("<html>");
    out.println("<head>");
    out.println("<title>登录成功</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>欢迎! "+name+"您已成功登录系统,.....,</h1>");
    out.println("</body>");
    out.println("</html>");

    out.close();
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}

// </editor-fold>
}

```

程序说明：作为登录验证成功的响应，显示一条欢迎信息。由于 `RequestDispatcher` 对象的 `forward` 方法将前端的请求对象 `request` 传递到本 `Servlet`，因此，依然可以调用 `request` 对象的 `getParameter("userID")` 方法来获取用户的登录 ID。`request` 对象的生命周期直到服务器端向客户端返回响应时才宣告结束。

程序 2-17: LoginFail.java

```

package example.servlet;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class LoginFail extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
    }
}

```

```

        out.println("<title> 登录失败</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>登录失败, 请重新登录.....</h1>");
        RequestDispatcher dispatcher = request.getRequestDispatcher("login.html");
        dispatcher.include(request, response);
        out.println("</body>");
        out.println("</html>");
        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    public String getServletInfo() {
        return "Short description";
    }
}
// </editor-fold>
}

```

程序说明：作为登录验证失败的响应，显示一条错误提示信息。并调用 `RequestDispatcher` 对象的 `include()` 方法，将登录页面作为响应的一部分输出到客户端显示。

运行结果

程序发布成功后，在浏览器地址栏输入“`http://localhost:8080/Servlet/login.html`”，将得到如图 2-14 所示的运行结果。



图 2-14 系统登录页面

输入用户名和密码信息后，单击【提交】按钮，登录信息发送到后端 `Servlet` 组件 `Main` 进行验证（为使代码简洁，本例中假定用户名和密码皆为 `guest`）。如果登录信息正确，则得到如图 2-15 所示的成功登录提示，否则，将得到如图 2-16 所示的登录失败提示。



图 2-15 登录成功提示



图 2-16 登录失败提示

讨论与思考

如果服务器端对客户的响应比较复杂, 往往需要多个服务器端组件一起协同工作才能够完成, 服务器端组件协同的最根本方法即是创建请求指派对象 `RequestDispatcher`, 调用 `RequestDispatcher` 的 `forward` 方法或 `include` 方法将客户端的请求和响应对象在服务器组件间传递。其中, `forward` 方法是将客户端请求传递到下一个组件进行处理, `include` 方法是将其其他服务器组件对客户端请求的处理和响应作为组件本身对服务器请求响应的一部分。

上述服务器组件之间的协作是在服务器上进行的, 客户端根本无从知晓。除了上面提到的方法外, 还有一种客户端重定向的方法, 即调用 `HttpServletResponse` 对象 `sendRedirect()` 方法, 将客户端请求重定位到一个新的组件, 但这种方式本质上是组件通知浏览器发起一次新的请求, 从客户端浏览器地址栏的变化即可证实, 因此, 这种重定位方法与上面介绍的服务器组件之间的协同有本质的区别。

知识点索引

`RequestDispatcher`: 重定位。

本章小结

Web 应用程序的基本工作模式是一种服务器和客户端之间的“请求-响应”模式, 即客户端浏览器

向服务器发出请求，服务器对客户端信息做出响应。因此 Java EE Web 应用开发的核心便是对服务器与浏览器之间请求响应的处理和控制。

Java EE 规范定义了 Java Servlet API，用于定义服务器和 Servlet 之间的标准接口。JSP 也是在编译后转换为 Servlet 并部署在服务器上运行的。Java Servlet API 是一组接口和类，主要由两个包组成：javax.servlet 包含了支持协议无关的 Servlet 的类和接口；javax.servlet.http 包括了对 HTTP 协议的特别支持的接口和类。如果希望详细了解 Java Servlet API，可以到网址“<http://java.sun.com/products/servlet/index.html>”下载 Java Servlet API 的详细文档。

所有的 Servlet 都必须实现下面的两个接口之一：通用 Servlet 接口和 HttpServlet 接口。通用 Servlet 接口类 javax.servlet.GenericServlet 定义了管理 Servlet 及它与客户机通信的方法。HttpServlet 接口类 javax.servlet.http.HttpServlet 是继承了通用 Servlet 接口类的一个抽象子类。要编写在 Web 上使用的 HTTP Servlet，通常采用继承 HTTP Servlet 接口类的形式。下面以 HttpServlet 接口类为中心，介绍与 Servlet 编程密切相关的几个接口，如图 2-17 所示。

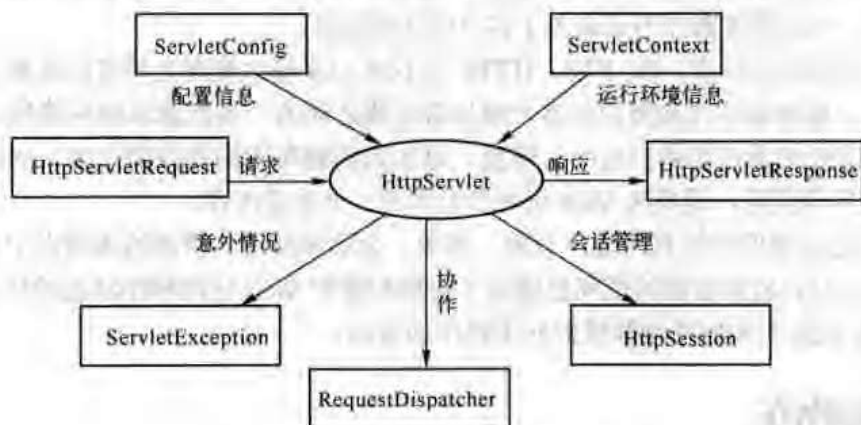


图 2-17 Servlet API 核心接口和类

HttpServletRequest：代表发送到 HTTP Servlet 的请求。这个接口封装了从客户机到服务器的通信。它可以包含关于客户机环境的信息和任何要从客户机发送到 Servlet 的数据。

HttpServletResponse：代表从 HTTP Servlet 发回客户机的响应。它通常是根据请求和 Servlet 访问的其他来源中的数据动态创建生成的响应，如 HTML 页面。

ServletConfig：代表 Servlet 的配置信息。Servlet 在发布到服务器上的时候，在 Web 应用配置文件中对应一段配置信息。Servlet 根据配置信息进行初始化。配置信息的好处在于在 Servlet 发布时可以通过配置信息灵活地调整 Servlet 而不需要重新改动、编译代码。

ServletContext：代表 Servlet 的运行环境信息。Servlet 是运行在服务器上的程序。为了与服务器及服务器上运行的其他程序进行交互，有必要获得服务器的环境信息。

ServletException：代表 Servlet 运行过程中掷出的意外对象。

HttpSession：用来在无状态的 HTTP 协议下越过多个请求页面来维持状态和识别用户。维护 HttpSession 的方法有 Cookie 或 URL 重写。

RequestDispatcher：请求转发器，可以将客户端请求从一个 Servlet 转发到其他的服务器资源，如其他 Servlet、静态 HTML 页面等。

服务器必须“声明”支持编写 Java Servlet API 的版本级别。目前 Java Servlet API 的最新版本为 2.5。通过以上几个类和接口，便可以很方便地实现对服务器和浏览器之间交互的控制。

本章还通过几个示例演示了如何实现服务器和浏览器之间的复杂交互，如发送非 HTML 文件、客户端“拉”技术、过滤器等。

第3章 管理和维护应用程序状态

使人者，器之。

——《论语·颜渊》

当前 Internet 上占统治地位的应用层协议是 HTTP 协议。HTTP 协议是一种无状态的协议，客户端每次打开一个 Web 页面，它就会与服务器建立一个新的连接，发送一个新的请求到服务器，服务器处理客户端的请求，返回响应到客户端，并关闭与客户端建立的连接。当客户端发起新的请求，那么它重新与服务器建立连接，因此服务器并不记录关于客户的任何信息。

相对于保持连接的通信协议，如 FTP，HTTP 协议可以在很大程度上节省服务器的资源，但是对于许多 Web 应用而言，服务器往往需要记录客户端与服务器之间的一系列请求响应的相关特定信息。例如，一个在线网上商店需要记录在线客户的个人信息、添加到购物车中的商品信息等。因此，如何管理和维护 Web 应用程序的状态信息，就成为 Web 应用开发中的一个重要内容。

Web 应用中信息存储的空间有四类：页面、请求、会话和应用。管理的本质在于充分而有效地利用资源，因此如何将信息存放到合适的空间也就成了管理和维护 Web 应用程序状态的核心。本章将通过多个示例演示 Java EE 框架下如何管理和维护应用程序的状态。

例程 3-1：购物车

目的

演示在 Web 应用中如何实现会话跟踪。

问题

在一个在线销售的网上水果店中，购物车是必需的一个组件，它记录了客户在网站上的整个采购过程信息。如何利用 Java EE 技术实现购物车组件？

知识链接

从特定客户端到服务器的一系列请求称为会话。记录会话信息的技术称为会话跟踪。客户的采购信息跨越多个页面，是一种会话范围内的信息。因此记录客户采购信息必须采用会话跟踪技术。

对于程序员而言会话跟踪不是容易解决的问题。会话跟踪的第一个障碍是如何唯一标识每一个客户会话。这只能通过为每一个客户分配一个某种标识，并将这些标识保存在客户端上，以后客户端发给服务器的每一个 HTTP 请求都提供这些标识来实现。

常见会话跟踪技术有：Cookie、URL 重写和隐藏表单域。Servlet 技术提供了 HttpSession 对象来帮助开发人员实现会话跟踪。

解决方案 1: 利用隐藏字段实现购物车

将客户采购信息保存在表单的隐藏字段中, 连同表单的其他信息一起提交到服务器, 实现会话跟踪功能。

实现步骤

- (1) 创建水果清单列表组件 FruitCatalog.java;
- (2) 创建购物车组件 hShoppingCart.java。

示例代码

本例程的所有示例代码均在 netbeans 的工程 Servlet 内。

程序 3-1: FruitCatalog.java

```
package example.servlet;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class FruitCatalog extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=gb2312");
        PrintWriter out = response.getWriter();
        out.println("<HTML><HEAD><TITLE>简单购物车 "
            + "Example</TITLE></HEAD>");
        out.println("<BODY><TABLE border=\"0\" width=\"100%\"><tr>");
        out.println("<td align=\"left\" valign=\"bottom\">");
        out.println("<H1>欢迎来到网上水果店购物</H1></td></tr></TABLE><HR>");
        out.print("<FORM ACTION=\"");
        out.println(response.encodeURL("hShoppingCart"));
        out.println("\" METHOD=\"POST\">");
        out.println("<TABLE CELLSPACING=\"5\" CELLPADDING=\"5\"><TR>");
        out.println("<TD ALIGN=\"center\"><B>种类</B></TD>");
        out.println("<TD ALIGN=\"center\"><B>单价</B></TD>");
        out.println("<TD ALIGN=\"center\"><B>数量</B></TD></TR><TR>");
        out.println("<TD ALIGN=\"center\">+\"苹果\"+</TD>");
        out.println("<TD ALIGN=\"center\">+\"6.5\"+</TD>");
        out.println("<TD ALIGN=\"center\">");
        out.println("<INPUT NAME=\"apple_amount\" "
            + "></TD></TR><TR>");
        out.println("<TD ALIGN=\"center\">+\"香蕉\"+</TD>");
        out.println("<TD ALIGN=\"center\">+\"3.5\"+</TD>");
        out.println("<TD ALIGN=\"center\">");
        out.println("<INPUT NAME=\"banana_amount\"
```



```

        + "></TD></TR><TR>";
    out.println("<TD ALIGN=\"center\">\"葡萄\"</TD>");
    out.println("<TD ALIGN=\"center\">\"5.6\"</TD>");
    out.println("<TD ALIGN=\"center\">");
    out.println("<INPUT NAME=\"grape_amount\"\"");
        + "></TD></TR>");
    out.println("</TABLE><HR>");
    String apple_amount = request.getParameter("AppleAmount");
    String banana_amount = request.getParameter("BananaAmount");
    String grape_amount = request.getParameter("GrapeAmount");
    if(apple_amount==null)apple_amount="0";
    if(banana_amount==null)banana_amount="0";
    if(grape_amount==null)grape_amount="0";
    out.println("<INPUT TYPE=HIDDEN NAME=\"AppleAmount\" VALUE=\"\" + apple_amount +");
        + "\">");
    out.println("<INPUT TYPE=HIDDEN NAME=\"BananaAmount\" VALUE=\"\" + banana_amount");
        + "\">");
    out.println("<INPUT TYPE=HIDDEN NAME=\"GrapeAmount\" VALUE=\"\" +grape_amount +");
        + "\">");
    out.println("<INPUT TYPE=\"Submit\" NAME=\"btn_submit\" \" + \"VALUE=\"放入");
        + "购物车\">");
    out.println("</FORM></BODY></HTML>");
    out.close();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}
}

```

程序说明：程序主要显示销售水果的基本信息，供顾客进行选择。同时，调用 request 对象的 getParameter() 方法获取客户已经进行的采购信息，并将其放置到表单的隐藏字段中。

程序 3-2: hShoppingCart.java

```

package example.servlet;

import java.io.*;
import java.net.*;

import javax.servlet.*;

```

```

import javax.servlet.http.*;
public class hShoppingCart extends HttpServlet {

    protected void processRequest(HttpServletRequest req, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        String apple_amount=req.getParameter("apple_amount");
        String banana_amount=req.getParameter("banana_amount");
        String grape_amount=req.getParameter("grape_amount");
        String appleAmount=req.getParameter("AppleAmount");
        String bananaAmount=req.getParameter("BananaAmount");
        String grapeAmount=req.getParameter("GrapeAmount");

        int new_apple_amount =Integer.parseInt(appleAmount,10)+Integer.parseInt(
            (apple_amount,10));
        int new_banana_amount =Integer.parseInt(bananaAmount,10)+Integer.parseInt(
            (banana_amount,10));
        int new_grape_amount =Integer.parseInt(grapeAmount,10)+Integer.parseInt(
            (grape_amount,10));

        // Print Current Contents of Cart
        out.println("<HTML><HEAD><TITLE>");
        out.println("购物车内容");
        out.println("</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("<H1>你的购物车里有</H1>"+
            new_apple_amount+"斤苹果 "+new_banana_amount+"斤香蕉 "
            +new_grape_amount+"斤葡萄");
        out.println("<HR>");
        out.println("<FORM ACTION=\"FruitCatalog\" METHOD=POST>");
        out.println("<INPUT TYPE=HIDDEN NAME=\"AppleAmount\" VALUE=\"\" +
            new_apple_amount + \">");
        out.println("<INPUT TYPE=HIDDEN NAME=\"BananaAmount\" VALUE=\"\" +
            new_banana_amount + \">");
        out.println("<INPUT TYPE=HIDDEN NAME=\"GrapeAmount\" VALUE=\"\" +
            new_grape_amount + \">");

        out.println("<INPUT TYPE=SUBMIT VALUE=\" 返回继续购物 \">");
        out.println("</FORM>");
        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)

```



```

throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}
}

```

程序说明：程序基于隐藏字段实现了购物车功能。首先调用 `HttpRequest` 对象的 `getParameter()` 方法获取隐藏字段传递来的购物车信息，然后在页面显示购物车信息。为了还能够在返回的水果列表页面中保存购物会话信息，程序仍旧使用表单提交的方式向水果列表组件 `Servlet` 发出请求，并将购物会话信息以表单的隐藏字段的形式发送。

运行结果

程序发布成功后，在浏览器地址栏中输入“`http://localhost:8080/Servlet/FruitCatalog`”，将得到如图 3-1 所示的运行结果。

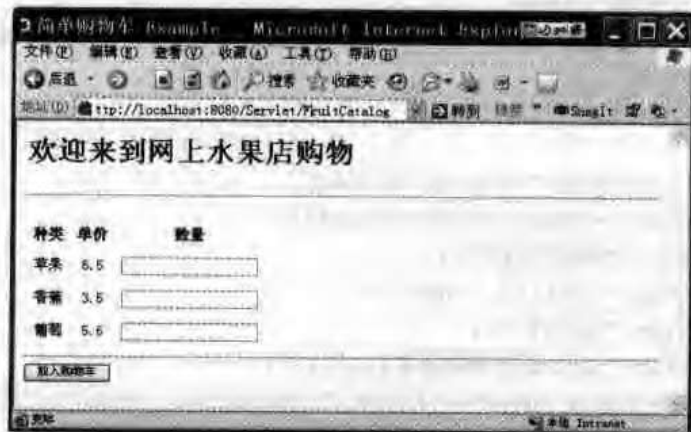


图 3-1 程序 3-1 运行结果画面

在文本输入框输入采购水果的数量后，单击【放入购物车】按钮，将得到如图 3-2 所示的运行结果。

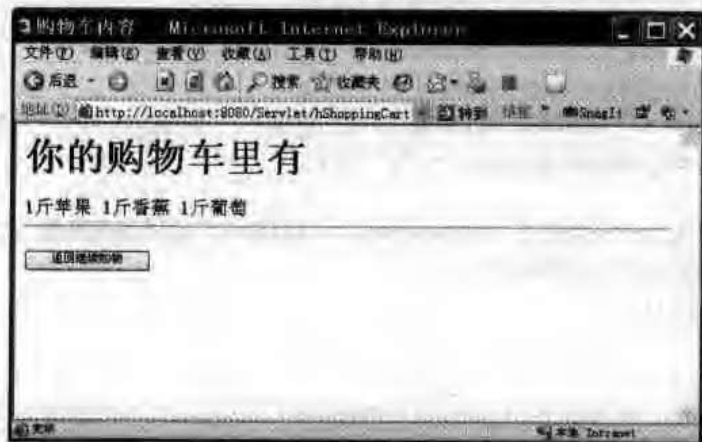


图 3-2 基于隐藏字段实现购物车

单击【返回继续购物】按钮，重新选购水果后将其添加到购物车，可以看到购物车组件成功累加了客户历次的购物信息。

解决方案2：利用 URL 重写实现购物车

将客户购物信息作为请求参数，以 URL 重写的方式在服务器和客户端浏览器之间传递。

实现步骤

- (1) 创建水果清单列表组件 uFruitCatalog.java;
- (2) 创建购物车组件 uShoppingCart.java。

示例代码

程序 3-3: uFruitCatalog.java

```
package example.servlet;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class uFruitCatalog extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=gb2312");
        PrintWriter out = response.getWriter();
        out.println("<HTML><HEAD><TITLE>简单购物车 "
            + "Example</TITLE></HEAD>");
        out.println("<BODY><TABLE border=\"0\" width=\"100%\"><tr>");
        out.println("<td align=\"left\" valign=\"bottom\">");
        out.println("<H1>欢迎来到网上水果店购物</H1></td></tr></TABLE><HR>");
        String apple_amount = request.getParameter("AppleAmount");
        String banana_amount = request.getParameter("BananaAmount");
        String grape_amount = request.getParameter("GrapeAmount");
        if(apple_amount==null)apple_amount="0";
        if(banana_amount==null)banana_amount="0";
        if(grape_amount==null)grape_amount="0";
        String encodedUrl = response.encodeURL("uShoppingCart?AppleAmount="
            +apple_amount+
            "&BananaAmount="+banana_amount+ "&GrapeAmount="+grape_amount);
        out.print("<FORM ACTION=\"");
        out.println(response.encodeURL(encodedUrl));
        out.println("\\" METHOD=\"POST\">");
        out.println("<TABLE CELLSPACING=\"5\" CELLPADDING=\"5\"><TR>");
        out.println("<TD ALIGN=\"center\"><B>种类</B></TD>");
        out.println("<TD ALIGN=\"center\"><B>单价</B></TD>");
        out.println("<TD ALIGN=\"center\"><B>数量</B></TD></TR><TR>");
        out.println("<TD ALIGN=\"center\">"+ "苹果" + "</TD>");
        out.println("<TD ALIGN=\"center\">"+ "6.5" + "</TD>");
```



```

        out.println("<TD ALIGN=\"center\">");
        out.println("<INPUT NAME=\"apple_amount\" "
            + "></TD></TR><TR>");
        out.println("<TD ALIGN=\"center\">\"香蕉\"</TD>");
        out.println("<TD ALIGN=\"center\">\"3.5\"</TD>");
        out.println("<TD ALIGN=\"center\">");
        out.println("<INPUT NAME=\"banana_amount\" "
            + "></TD></TR><TR>");
        out.println("<TD ALIGN=\"center\">\"葡萄\"</TD>");
        out.println("<TD ALIGN=\"center\">\"5.6\"</TD>");
        out.println("<TD ALIGN=\"center\">");
        out.println("<INPUT NAME=\"grape_amount\" "
            + "></TD></TR>");
        out.println("</TABLE><HR>");

        //String contextPath = request.getContextPath();

        out.println("<INPUT TYPE=\"Submit\" NAME=\"btn_submit\" " + "VALUE=\"放入购物车\">");
        out.println("</FORM></BODY></HTML>");

        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    public String getServletInfo() {
        return "Short description";
    }
}

```

程序说明：程序主要显示销售的水果的基本信息，供顾客进行选择。同时，调用 request 对象的 getParameter() 方法从请求 URL 中获取客户已经进行的采购信息。为实现会话跟踪，将购物会话信息组装成字符串并调用 HttpServletResponse 的 encodeURL() 方法进行编码，将编码后的字符串作为下次请求的 URL 地址，这样就实现了购物会话跟踪。

程序 3-4: uShoppingCart

```

package example.servlet;
import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

```

```

public class uShoppingCart extends HttpServlet {

    protected void processRequest(HttpServletRequest req, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=gb2312");
        PrintWriter out = response.getWriter();
        String apple_amount=req.getParameter("apple_amount");
        String banana_amount=req.getParameter("banana_amount");
        String grape_amount=req.getParameter("grape_amount");
        String appleAmount=req.getParameter("AppleAmount");
        String bananaAmount=req.getParameter("BananaAmount");
        String grapeAmount=req.getParameter("GrapeAmount");

        int new_apple_amount =Integer.parseInt(appleAmount,10)+Integer.parseInt(
            apple_amount,10);
        int new_banana_amount =Integer.parseInt(bananaAmount,10)+Integer.parseInt(
            banana_amount,10);
        int new_grape_amount =Integer.parseInt(grapeAmount,10)+Integer.parseInt(
            grape_amount,10);

        // Print Current Contents of Cart
        out.println("<HTML><HEAD><TITLE>");
        out.println("购物车内容");
        out.println("</TITLE></HEAD>");
        out.println("<BODY>");
        out.println("<H1>你的购物车里有</H1>"+
            new_apple_amount+"斤苹果 "+new_banana_amount+"斤香蕉 "+
            new_grape_amount+"斤葡萄");
        out.println("<HR>");
        String contextPath = req.getContextPath( );
        String encodedUrl = response.encodeURL(contextPath + "/uFruitCatalog?Apple
            Amount="+new_apple_amount+
            "&BananaAmount="+new_banana_amount+ "&GrapeAmount="+new_grape_amount);
        out.println("<a href=\""+ encodedUrl +
            "\">返回继续购物</a>.");
        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }
}

```



```
public String getServletInfo() {  
    return "Short description";  
}  
}
```

程序说明：由于购物会话信息以 URL 参数的形式传递过来，因此可以直接调用 `HttpRequest` 的 `getParameter()` 方法来获取购物会话信息。同时，为了向购物页面传递购物信息，也采用与程序同样的 URL 重写的方式实现。

运行结果

重新发布 Web 应用，在浏览器地址栏中输入“`http://localhost:8080/Servlet/uFruitCatalog`”，将得到如图 3-3 所示的运行结果。



图 3-3 程序 3-3 运行结果画面

在文本输入框输入采购水果的数量后，单击【放入购物车】按钮，将得到如图 3-4 所示的运行结果。



图 3-4 基于 URL 重写实现购物车

单击【返回继续购物】链接，重新选购水果后将其添加到购物车，可以看到购物车组件成功累加了客户历次的购物信息。

解决方案3：利用 Cookie 实现购物车

将购物会话信息放置在 Cookie 中，显示购物车内容时将信息从 Cookie 中取出。

实现步骤

- (1) 创建水果清单列表组件 cFruitCatalog.java;
- (2) 创建购物车组件 cShoppingCart.java。

示例代码

程序 3-5: cFruitCatalog.java

```
package example.servlet;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class cFruitCatalog extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=gb2312");
        PrintWriter out = response.getWriter();
        out.println("<HTML><HEAD><TITLE>简单购物车 "
            + "Example</TITLE></HEAD>");
        out.println("<BODY><TABLE border=\"0\" width=\"100%\"><tr>");
        out.println("<td align=\"left\" valign=\"bottom\">");
        out.println("<H1>欢迎来到网上水果店购物</H1></td></tr></TABLE><HR>");
        out.print("<FORM ACTION=\"");
        out.println(response.encodeURL("cShoppingCart"));
        out.println("\" METHOD=\"POST\">");
        out.println("<TABLE CELLSPACING=\"5\" CELLPADDING=\"5\"><TR>");
        out.println("<TD ALIGN=\"center\"><B>种类</B></TD>");
        out.println("<TD ALIGN=\"center\"><B>单价</B></TD>");
        out.println("<TD ALIGN=\"center\"><B>数量</B></TD></TR><TR>");
        out.println("<TD ALIGN=\"center\">\"+\"苹果\"+\"</TD>");
        out.println("<TD ALIGN=\"center\">\"+\"6.5\"+\"</TD>");
        out.println("<TD ALIGN=\"center\">");
        out.println("<INPUT NAME=\"apple_amount\"\"
            + \"></TD></TR><TR>");
        out.println("<TD ALIGN=\"center\">\"+\"香蕉\"+\"</TD>");
        out.println("<TD ALIGN=\"center\">\"+\"3.5\"+\"</TD>");
        out.println("<TD ALIGN=\"center\">");
        out.println("<INPUT NAME=\"banana_amount\"\"
            + \"></TD></TR><TR>");
        out.println("<TD ALIGN=\"center\">\"+\"葡萄\"+\"</TD>");
```



```

        out.println("<TD ALIGN=\"center\">"+5.6+"</TD>");
        out.println("<TD ALIGN=\"center\">");
        out.println("<INPUT NAME=\"grape_amount\" "
            + "></TD></TR>");
        out.println("</TABLE><HR>");

        out.println("<INPUT TYPE=\"Submit\" NAME=\"btn_submit\" " + "VALUE=\"放入购物车\">");
        out.println("</FORM></BODY></HTML>");
        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    public String getServletInfo() {
        return "Short description";
    }
}

```

程序说明：程序提供可选购的水果信息，并负责将信息提交到后端组件。

程序 3-6: cShoppingCart.java

```

package example.servlet;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class cShoppingCart extends HttpServlet {

    protected void processRequest(HttpServletRequest req, HttpServletResponse response)
        throws ServletException, IOException {
        //读取请求参数
        String apple_amount=req.getParameter("apple_amount");
        String banana_amount=req.getParameter("banana_amount");
        String grape_amount=req.getParameter("grape_amount");
        int appleN=Integer.parseInt(apple_amount,10);
        int bananaN=Integer.parseInt(banana_amount,10);
        int grapeN=Integer.parseInt(grape_amount,10);
        Cookie cookie = null;
        Cookie cookie1 = null;
    }
}

```

```
Cookie cookie2 = null;
//获取请求相关的Cookie
Cookie[] cookies = req.getCookies();
//判断Cookie[]
int total_apple=appleN;int total_banana=bananaN;int total_grape=grapeN;

if (cookies != null){
    for (int i = 0; i < cookies.length; i++){
        if (cookies[i].getName().indexOf("AppleAmount")!=-1){
            String v=cookies[i].getValue();
            int value=Integer.parseInt(v);
            total_apple+=value;
        }
        if (cookies[i].getName().indexOf("BananaAmount")!=-1){
            String v=cookies[i].getValue();
            int value=Integer.parseInt(v);
            total_banana+=value;
        }
        if (cookies[i].getName().indexOf("GrapeAmount")!=-1){
            String v=cookies[i].getValue();
            int value=Integer.parseInt(v);
            total_grape+=value;
        }
    }
}
//end for
}
//end if
//将新提交的数据以Cookie形式保存
long m=System.currentTimeMillis();
String mark=Long.toString(m);
if (cookie == null){
    int maxAge=-1;
    //Create the Cookie object
    cookie = new Cookie("AppleAmount"+mark,apple_amount);
    cookie.setPath(req.getContextPath());
    cookie.setMaxAge(maxAge);
    response.addCookie(cookie);
}
//end if
if (cookie1 == null){
    int maxAge=-1;
    //Create the Cookie object
    cookie1 = new Cookie("BananaAmount"+mark,banana_amount);
    cookie1.setPath(req.getContextPath());
    cookie1.setMaxAge(maxAge);
    response.addCookie(cookie1);
}
//end if
if (cookie2 == null){
```



```

        int maxAge=-1;
        //Create the Cookie object
        cookie2 = new Cookie("GrapeAmount"+mark,grape_amount);
        cookie2.setPath(req.getContextPath( ));
        cookie2.setMaxAge(maxAge);
        response.addCookie(cookie2);
    } //end if

    response.setContentType("text/html;charset=gb2312");
    PrintWriter out = response.getWriter();
    out.println("<HTML><HEAD><TITLE>");
    out.println("购物车内容");
    out.println("</TITLE></HEAD>");
    out.println("<BODY>");
    out.println("<H1>你的购物车里有</H1>"+
        total_apple+"斤苹果 "+total_banana+"斤香蕉 "
        +total_grape+"斤葡萄");
    out.println("<HR>");
    String contextPath = req.getContextPath( );
    String encodedUrl = response.encodeURL(contextPath + "/cFruitCatalog");
    out.println("<a href=\""+ encodedUrl +
        "\">返回继续购物</a>.");
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}
}

```

程序说明：程序调用 `HttpRequest` 的 `getParameter()` 方法获取提交的购物信息，并调用 `HttpRequest` 的 `getCookies()` 方法获取与请求相关的 `Cookie` 对象。每一次的购物信息都存储在一个相关的 `Cookie` 内。在本例中，保存苹果购物信息的 `Cookie` 将以“AppleAmount”加一个时间戳作为名字，以区别保存不同次别购物信息。为保证 `Cookie` 在会话结束后失效，还要将 `Cookie` 的生命周期设为-1。

运行结果

重新发布应用程序，在浏览器地址栏中输入“<http://localhost:8080/Servlet/uFruitCatalog>”，将得到如

图 3-5 所示的运行结果。



图 3-5 程序 3-5 运行结果画面

在文本输入框输入采购水果的数量后，单击【放入购物车】按钮，将得到如图 3-6 所示的运行结果。



图 3-6 基于 Cookie 实现购物车

单击【返回继续购物】链接，重新选购水果后将其添加到购物车，可以看到购物车组件成功累加了客户历次的购物信息。

解决方案 4：利用 Session 实现购物车

Servlet API 规范定义了一个简单的 HttpSession 接口，通过它开发人员可以方便地实现会话跟踪。

HttpSession 接口提供了一种把对象保存到内存、在同一用户的后继请求中提取这些对象的标准办法。例如，在会话中保存数据的方法是 `setAttribute(String s, Object o)`，从会话提取原来所保存对象的方法是 `getAttribute(String s)`，要获得 HttpSession 对象，可以调用 `HttpServletRequest` 对象的 `getSession()` 方法。

通过将购物信息保存在 Servlet 的 HttpSession 对象内，可以很方便地实现会话跟踪功能。

实现步骤

- (1) 创建水果清单列表组件 FruitCatalog.java;
- (2) 创建购物车组件 hShoppingCart.java。

示例代码

程序 3-7: sFruitCatalog.java

```
import java.io.*;
import java.util.HashMap;

import javax.servlet.*;
import javax.servlet.http.*;

public class sFruitCatalog extends HttpServlet {

    protected void processRequest(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        res.setContentType("text/html;charset=gb2312");
        PrintWriter out = res.getWriter();
        out.println("<HTML><HEAD><TITLE>简单购物车 "
            + "Example</TITLE></HEAD>");
        out.println("<BODY><TABLE border=\"0\" width=\"100%\"><tr>");
        out.println("<td align=\"left\" valign=\"bottom\">");
        out.println("<H1>欢迎来到网上水果店购物</H1></td></tr></TABLE><HR>");
        out.print("<FORM ACTION=\"");
        out.println(res.encodeURL("sShoppingCart"));
        out.println("\" METHOD=\"POST\">");
        out.println("<TABLE CELLSPACING=\"5\" CELLPADDING=\"5\"><TR>");
        out.println("<TD ALIGN=\"center\"><B>种类</B></TD>");
        out.println("<TD ALIGN=\"center\"><B>单价</B></TD>");
        out.println("<TD ALIGN=\"center\"><B>数量</B></TD></TR><TR>");
        out.println("<TD ALIGN=\"center\">\"+\"苹果\"+\"</TD>");
        out.println("<TD ALIGN=\"center\">\"+\"6.5\"+\"</TD>");
        out.println("<TD ALIGN=\"center\">");
        out.println("<INPUT NAME=\"apple_amount\"\"
            + \"></TD></TR><TR>");
        out.println("<TD ALIGN=\"center\">\"+\"香蕉\"+\"</TD>");
        out.println("<TD ALIGN=\"center\">\"+\"3.5\"+\"</TD>");
        out.println("<TD ALIGN=\"center\">");
        out.println("<INPUT NAME=\"banana_amount\"\"
            + \"></TD></TR><TR>");
        out.println("<TD ALIGN=\"center\">\"+\"葡萄\"+\"</TD>");
        out.println("<TD ALIGN=\"center\">\"+\"5.6\"+\"</TD>");
        out.println("<TD ALIGN=\"center\">");
        out.println("<INPUT NAME=\"grape_amount\"\"
            + \"></TD></TR>");
        out.println("</TABLE><HR>");
        out.println("<INPUT TYPE=\"Submit\" NAME=\"btn_submit\" \"
            + \"VALUE=\"放入购物车\">");
    }
}
```

```

        out.println("</FORM></BODY></HTML>");
        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    public String getServletInfo() {
        return "Short description";
    }
}

```

程序说明：程序提供可选购的水果信息，并负责将信息提交到后端组件。

程序 3-8: sShoppingCart.java

```

package example.servlet;

import java.io.*;
import java.util.HashMap;
import javax.servlet.*;
import javax.servlet.http.*;

public class sShoppingCart extends HttpServlet {
    protected void processRequest(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
        HttpSession session = req.getSession(true);
        HashMap cart = (HashMap) session.getAttribute("cart");
        if (cart == null) {
            cart = new HashMap();
            cart.put("apple", "0");
            cart.put("banana", "0");
            cart.put("grape", "0");
            session.setAttribute("cart", cart);
        }
        res.setContentType("text/html;charset=gb2312");
        PrintWriter out = res.getWriter();
        req.setCharacterEncoding("gb2312");
        String apple_amount=req.getParameter("apple_amount");
        String banana_amount=req.getParameter("banana_amount");
        String grape_amount=req.getParameter("grape_amount");
        String appleAmount=(String)cart.get("apple");
        String bananaAmount=(String)cart.get("banana");
        String grapeAmount=(String)cart.get("grape");
        //int te=Integer.parseInt(appleAmount,10);
        //int tel=Integer.getInteger(apple_amount).intValue();
    }
}

```



```

int new_apple_amount =Integer.parseInt(appleAmount,10)+Integer.parseInt
(apple_amount,10);
int new_banana_amount =Integer.parseInt(bananaAmount,10)+Integer.parseInt
(banana_amount,10);
int new_grape_amount =Integer.parseInt(grapeAmount,10)+Integer.parseInt
(grape_amount,10);
cart.put("apple",String.valueOf(new_apple_amount));
cart.put("banana",String.valueOf(new_banana_amount));
cart.put("grape",String.valueOf(new_grape_amount));
// Print Current Contents of Cart
out.println("<HTML><HEAD><TITLE>");
out.println("购物车内容");
out.println("</TITLE></HEAD>");
out.println("<BODY>");
out.println("<H1>你的购物车里有</H1>"+
    new_apple_amount+"斤苹果 "+new_banana_amount+"斤香蕉 "
    +new_grape_amount+"斤葡萄");
out.println("<HR>");
out.print("<HR><p><A HREF=\"");
out.print(res.encodeURL("sFruitCatalog"))");
out.println(">回到水果店</A></p>");
out.close();

}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}
}

```

程序说明：程序首先从会话中获取名为 `cart` 的 `HttpSession` 对象，如果此对象不存在，则创建一个 `Hasnmap` 对象，并把此对象命名为 `cart` 保存到会话中。根据客户提交的请求信息，获取各种水果的数量，来更新 `session` 对象 `cart`，并将此对象重新保存到会话中。

运行结果

重新发布应用程序，在浏览器地址栏中输入“<http://localhost:8080/Servlet/sFruitCatalog>”，将得到如图 3-7 所示的运行结果。

在文本输入框输入采购水果的数量后，单击【放入购物车】按钮，将得到如图 3-8 所示的运行结果。

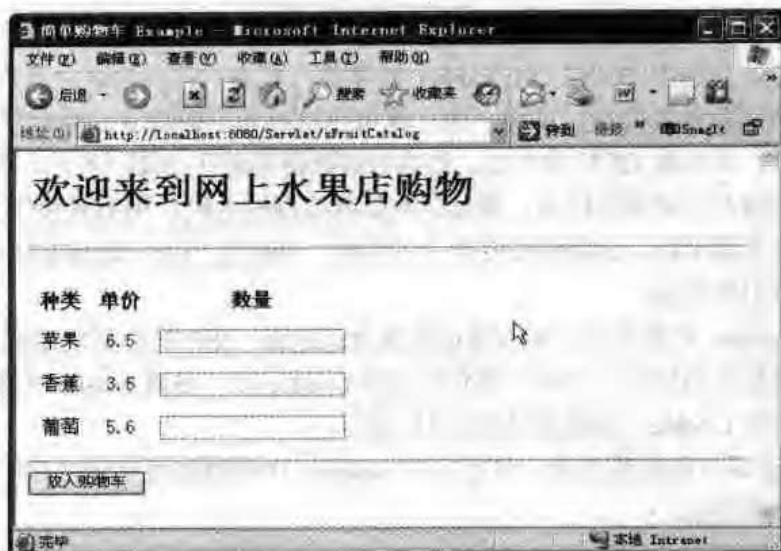


图 3-7 程序 3-7 运行结果画面



图 3-8 基于会话实现购物车

单击【返回继续购物】链接，重新选购水果后将其添加到购物车，可以看到购物车组件成功累加了客户历次的购物信息。

讨论与思考

在本例程中，列举了目前几乎所有的会话跟踪方法，其中每种方法都有其优缺点和适用场景。

Cookie 是一个小小的文本文件，用于记录会话信息，每个页面都去 Cookie 中提取以前的会话信息。虽然 Cookie 强大且持续性高，但是由于有些用户因为担心 Cookie 对个人隐私的威胁，会关闭 Cookie，一旦如此，便无法利用 Cookie 来实现会话追踪的功能。

URL 重写是利用 GET 的方法，在 URL 的尾部添加一些额外的参数来达到会话追踪的目的，服务器将这个标识符与它所存储的有关会话的数据关联起来。

使用 URL 重写的优点是，在 Cookie 被禁用或者根本不支持的情况下依旧能够工作。但也有很多缺点：

- ❖ 必须对所有指向您的网站的 URL 进行编码；
- ❖ 所有页面必须动态生成；

◇ 不能使用预先记录下来的 URL 进行访问，或者从其他网站链接进行访问。

隐藏表单字段的方法是利用 HTML 内 hidden 的属性，把客户端的信息在用户不察觉的情形下偷偷地随着请求一起传送到服务器处理，这样一来，就可以进行会话跟踪了。隐藏字段的优点在于，会话数据传送到服务器端时，并不像 GET 的方法，会将会话数据暴露在 URL 之上。不过这种做法还是有它的缺点：一旦会话数据储存在隐藏字段中，就仍然有暴露数据的危机。因为只要用户直接观看 HTML 的源文件，会话数据将会暴露无疑。这将造成安全上的漏洞，特别是当用户数据是依赖于用户 ID、密码取得的时候，将会有被盗用的危险。

JSP 使用内建的 session 对象可以非常方便地实现会话追踪，JSP 的会话机制基于 Cookie 或 URL 重写技术，融合了这两种技术的优点，当客户端允许使用 Cookie 时，内建 session 对象使用 Cookie 进行会话追踪，如果客户端禁用 Cookie，则选择使用 URL 重写。

session 的优点是会话信息管理方便。但是由于 session 中的数据存储在内存中，如果占用的空间较大，则会影响到服务器的性能。

知识点索引

会话跟踪；表单处理；URL 重写；Cookie；HttpSession。

例程 3-2：聊天室

目的

- (1) 演示如何维护管理应用程序运行过程中的状态信息；
- (2) 利用监听器监视应用状态变化；
- (3) 利用框架技术实现复杂 Web 页面。

问题

聊天室是不少网站建设中常见的组件。要求实现一个具有以下功能的聊天室：

- (1) 多人共同聊天；
- (2) 防止恶意刷屏；
- (3) 语言过滤功能，避免一些敏感词汇；
- (4) 显示当前聊天客户列表；
- (5) 显示聊天室人员变动信息（进入聊天室、离开聊天室等）。

解决方案

利用 JSP 和 Servlet 实现聊天室程序。JSP 显示聊天界面，Servlet 实现后台的服务和管理。信息存放的范围有 request，代表客户端对服务器的一次请求；session 代表客户端与服务器的交互过程；application 代表整个 Web 应用程序。聊天室程序中涉及的大量信息需要管理维护，如所有聊天者的交谈信息、聊天用户列表、聊天用户个人信息，客户当前发言等，它们分别应该存放在适当的范围内。通过 Servlet 的监听器接口实现对 Web 应用的生命周期、属性变化，以及 session 的生命周期和属性变化事件的监听。

实现步骤

- (1) 生成代表聊天信息的辅助工具类 MessageQueue;
- (2) 利用框架技术生成聊天界面相关的页面;
- (3) 生成处理聊天信息的后端 Servlet 组件;
- (4) 利用 Web 应用上下文监听器监听聊天室启动信息,以便在聊天界面显示欢迎信息;
- (5) 利用会话监听器实现用户退出聊天室时的公告信息发布;
- (6) 利用会话属性监听器实现用户加入聊天室时的公告信息发布。

示例代码

本例程的所有示例代码均在 netbeans 的工程 Mychat 内。

程序 3-9: index.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>聊天室</title>
  </head>
  <body>

    <h1>欢迎来到聊天室</h1>
    <form name="form1" method="get" action="LoginServlet">
      <p align="left">请输入您的昵称: </p>
      <p align="left">
        <input type="text" name="user_id" size="10" maxlength="20">
        <input type="submit" name="Submit" value="确定">
      </p>
    </form>
  </body>
</html>
```

程序说明: 程序提供一个登录界面, 供聊天客户登录。聊天客户提交的信息将发往后端的组件 LoginServlet 进行处理。

程序 3-10: chatCenter.html

```
<html>
<head>
<title>简易聊天室</title>
</head>
<frameset rows="*,70" frameborder="NO" border="0" framespacing="0">
  <frameset cols="*,120" frameborder="NO" border="0" framespacing="0">
    <frame name="mainFrame" src="ChatViewer.jsp" >
    <frame name="rightFrame" noresize src="UserList.jsp" >
  </frameset>
  <frame name="bottomFrame" scrolling="NO" noresize src="talk.jsp" >
</frameset>
```



```

</body>
</html>

```

程序说明：程序实现整个聊天界面的总体框架。对于复杂的界面，采用一个单独的页面显示往往是不够的，经常需要将多个页面利用框架组合起来共同显示。聊天室界面就是由三个页面组合而成，左上显示当前聊天内容的组件 ChatViewwe.jsp，右上显示当前聊天室的用户列表组件 UserList.jsp，底部显示聊天输入界面。

程序 3-11: talk.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>聊天室</title>
    </head>
    <body>

        <form name="form1" method="get" action="ChatServlet" >
<p>
        <b>
            <font color="#CC9900">
                <%=session.getAttribute("ID")%> :
            </font>
        </b>
        <input type="text" name="message" size="30" maxlength="30">
        <input type="submit" name="Submit" value="确定">
        <input type="button" name="exit" value="退出聊天室" onclick="javascript:
            window.open('LoginServlet?action=logout','_parent')">
        </p>
        </form>

    </body>
</html>

```

程序说明：程序显示聊天界面。它主要提供两个功能：输入聊天信息和退出聊天室。在退出聊天室时，采用了 JavaScript 脚本与服务器端交互。因为当前页面嵌套在框架内部，因此在 JavaScript 脚本中采用 _parent 参数使得新打开的页面显示在上层的主框架内。

程序 3-12: ChatViewer.jsp

```

<%@ page language="java" %>
<%@ page import="java.lang.*,beans.MessageQueue" %>
<%@ page contentType="text/html; charset=gb2312" %>
<%
    response.setHeader( "Refresh" ,"5" ) ;
%>
<html>
<head>
<title>聊天室</title>
</head>
<body bgcolor="#FFFFFF">

```

```

<%MessageQueue mq=(MessageQueue)application.getAttribute("MESSAGE");
if(mq==null)mq=new MessageQueue();
%>
<%=mq.getMessage()%>
</body>
</html>

```

程序说明：程序用来显示聊天信息。所有的聊天信息都被封装在一个 MessageQueue 内，由于聊天室内的所有信息都需要看到此信息，因此 MessageQueue 必须添加到 application 内。另外，为了使页面自动刷新以加载新的聊天信息，调用 response 的 setHeader("Refresh", "5") 方法设置页面的文件头信息，使得页面实现 5 秒一次的定时刷新。

程序 3-13: UserList.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ page import="java.util.*" %>
<%
response.setHeader( "Refresh" , "5" ) ;
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%
HashMap users=(HashMap)this.getServletContext().getAttribute("USERS");
if(users==null)users=new HashMap();
%>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>JSP Page</title>
</head>
<body bgcolor="#FFCCCC">

<h5>人员列表</h5>
在线 <%=users.size()%>人<br>
<%
if(users!=null){
Iterator userlist=users.keySet().iterator();
while(userlist.hasNext()){
String nickname=(String)users.get(userlist.next());%>
<%=nickname%> <br>
<%
}
}
%>
</body>
</html>

```

程序说明：程序用来显示聊天客户列表。聊天客户列表存放在一个 HashMap 中，由于此信息所有聊

天用户都需要看到,因此 `HashMap` 对象被存放在 `application` 中。要遍历 `HashMap` 中的信息,首先将其转换为一个 `Iterator`,然后遍历此枚举对象即可。另外,为了使页面自动刷新以加载新的聊天信息,调用 `response` 的 `setHeader("Refresh", "5")` 方法设置页面的文件头信息,使得页面实现 5 秒一次的定时刷新。

程序 3-14: `MessageQueue.java`

```
package beans;

import java.util.Enumeration;
import java.util.Vector;
public class MessageQueue {

    private Object lockKey ;
    private Vector msgQueue ;
    private int maxNumMessages ;

    public MessageQueue() {
        this.lockKey=new Object() ;
        this.msgQueue=new Vector() ;
        this.maxNumMessages=10 ;
    }
    public String getMessages() {
        String htmlCode="" ;
        synchronized( lockKey ) {
            for( Enumeration e=this.msgQueue.elements() ; e.hasMoreElements() ; )
                htmlCode+=( "<p>"+(String)e.nextElement()+"</p>\n" ) ;
        }
        return htmlCode ;
    }
    public void setMaxNumMessages( String num ) {
        try {
            this.maxNumMessages=Integer.parseInt( num ) ;
        }
        catch( NumberFormatException ex ) {
            this.maxNumMessages=10 ;
        }
    }
    public void setMaxNumMessages( int num ) {
        this.maxNumMessages=num ;
    }
    public void setMessage( String msg ) {
        synchronized( lockKey ) {
            this.msgQueue.insertElementAt( msg, 0 ) ;
            int numMessages=this.msgQueue.size() ;
            if( numMessages>this.maxNumMessages )
                this.msgQueue.removeElementAt( numMessages-1 ) ;
        }
    }
}
```

程序说明: 程序作为一个辅助类,实现对聊天信息的抽象。`Servlet` 是支持多线程的,由于对象的实

体保存在 `application` 中，为了避免由于多线程访问而造成的错误，系统使用了“同步锁”机制，只有获得唯一的钥匙的线程才能对它进行修改。

程序 3-15: LoginServlet.java

```
package servlet;

import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class LoginServlet extends HttpServlet {

    HashMap users=null;

    static final int MaxUserNumber=5;//最大保存 50 条会话信息
    protected void processRequest (HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        //对于退出聊天室请求的处理
        String action=request.getParameter("action");
        if(action!=null){
            if(action.endsWith("logout")){
                //从用户列表中删除用户
                /*    users=(HashMap)this.getServletContext().getAttribute("USERS");
                if(users!=null)
                    users.remove(request.getSession().getId());*/
                request.getSession().invalidate();
            }
            RequestDispatcher dispatcher =
                getServletContext().getRequestDispatcher("/index.jsp");
            dispatcher.forward(request, response);

            return;
        }

        ///////////////////////////////////////////////////
        String id=request.getParameter("user_id");
        if(id==null){
            request.setAttribute("ERROR_Message","用户名不得为空...");
            RequestDispatcher dispatcher =
                getServletContext().getRequestDispatcher("/index.jsp");
            dispatcher.forward(request, response);
        }
        users=(HashMap)this.getServletContext().getAttribute("USERS");
        boolean flag=true;
        if(users!=null){
            Iterator userlist=users.keySet().iterator();
            while(userlist.hasNext()){
                String nickname=(String)users.get(userlist.next());
                if(nickname.equals(id)){flag=false;break;}
            }
        }
    }
}
```



```

    }
    }
    else users=new HashMap();

    if(flag){

        users.put(request.getSession().getId(),id);
        this.getServletContext().setAttribute("USERS",users);
        request.getSession().setAttribute("ID",id);//将用户 ID 放入会话中
        RequestDispatcher dispatcher =
            getServletConfig().getServletContext().getRequestDispatcher("/ChatCenter.html");
        dispatcher.forward(request, response);
    } else{
        request.setAttribute("ERROR_Message","用户名重复, 请选择新的用户名...");
        RequestDispatcher dispatcher =
            getServletConfig().getServletContext().getRequestDispatcher("/Index.jsp");
        dispatcher.forward(request, response);
    }

}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}
}

```

程序说明：程序实现聊天客户的登录和退出管理。主要代码都在方法 `processRequest()` 中。它首先根据客户请求的参数“action”判断用户请求的类型。如果是退出请求，则直接调用 `request` 对象的 `getSession()` 获得会话对象，然后调用会话对象的 `invalidate()` 方法使会话无效。随后利用 `RequestDispatcher` 对象转到登录页面。如果是登录请求，则首先检查当前聊天的用户列表中是否存在此名称的用户，如果存在，导向登录界面重新登录，如果不存在，则将当前用户信息加入 `application` 中的保存用户列表的 `HashMap` 对象，将用户名称信息加入用户会话，最终显示聊天界面。

程序 3-16: ChatServlet.java

```

package servlet;

import beans.MessageQueue;
import java.io.*;
import java.text.DateFormat;

```

```

import java.util.ArrayList;
import java.util.Date;

import javax.servlet.*;
import javax.servlet.http.*;

public class ChatServlet extends HttpServlet {
    static final int MaxmessageNumber=20;//最大保存 50 条会话信息

    protected void processRequest (HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        //检查是否在黑名单中
        if(!InBlackList( request)){
            String msg=request.getParameter( "message" );
            msg=preventFresh(request,msg);//防止刷屏
            msg=Check(msg);
            DateFormat dateFormat=DateFormat.getDateTimeInstance( DateFormat.SHORT,
            DateFormat.SHORT );
            String fullMessage="["+dateFormat.format( new Date() )+"]"
                +request.getSession().getAttribute("ID")+":&nbsp;"
                +msg;
            MessageQueue mq=(MessageQueue)this.getServletContext().getAttribute("MESSAGE");
            if(mq==null){mq=new MessageQueue();
            mq.setMaxNumMessages(MaxmessageNumber);
            }
            mq.setMessage(fullMessage);
            this.getServletContext().setAttribute("MESSAGE",mq);
            RequestDispatcher dispatcher =
                getServletConfig().getServletContext().getRequestDispatcher("/talk.jsp");
            dispatcher.forward(request, response);
        }
    }

    private String Check(String msg){

        //加入语言屏蔽功能
        ArrayList words =new ArrayList();
        words.add("狗屎");
        words.add("放屁");
        words.add("他妈的");
        words.add("牛B");
        words.add("滚犊子");
        words.add("法轮");
        boolean isFobidden=false;
        for(int i=0;i<words.size();i++){
            if(msg.indexOf((String) (words.get(i)))!=-1){
                isFobidden=true;
                break;
            }
        }
        if(isFobidden)msg="发言被禁止! 请使用文明语言....";
        return msg;
    }
}

```



```

    }
    private String preventFresh(HttpServletRequest request,String s){
        if( s==null || s.trim().length()==0 ) {

            s="" ;
        }
        //两次视为重复; 三次视为恶意刷屏
        String lastMsg=(String)request.getSession().getAttribute("LASTMSG");

        if(lastMsg==null||!lastMsg.equals(s)){//不重复

            request.getSession().setAttribute("LASTMSG",s);
            request.getSession().setAttribute("LASTLASTMSG",lastMsg);
            return s;
        }else{
            String lastlastMsg=(String)request.getSession().getAttribute("LASTLASTMSG");
            if(lastlastMsg==s){//三次重复
                //加入黑名单
                String ID=(String)request.getSession().getAttribute("ID");
                ArrayList blacklist=(ArrayList)this.getServletContext().getAttribute(
                    "BLACK_LIST");
                if(blacklist==null)blacklist=new ArrayList();
                blacklist.add(ID);
                this.getServletContext().setAttribute("BLACK_LIST",blacklist);
                return "由于多次重复发言刷屏, 您已被加入黑名单! ";
            }
            else{//两次重复
                request.getSession().setAttribute("LASTMSG",s);
                request.getSession().setAttribute("LASTLASTMSG",lastMsg);
                return " 禁止重复发言";
            }
        }

        //return false;
    }

    private boolean InBlackList(HttpServletRequest request){
        boolean isFobidden=false;
        ArrayList blacklist=(ArrayList)this.getServletContext().getAttribute(
            "BLACK_LIST");
        if(blacklist==null)return false;
        String ID=(String)request.getSession().getAttribute("ID");
        for(int i=0;i<blacklist.size();i++){
            if(blacklist.get(i).equals(ID)){
                isFobidden=true;
                break;
            }
        }
        return isFobidden ;
    }
}

```

```

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}
}

```

程序说明：实现聊天功能的主要组件，负责将处理用户提交的聊天信息，并将处理后的信息添加到 application 中的 MessageQueue 对象。方法 Check() 实现了对聊天信息的过滤功能，它主要利用字符串函数对客户提交信息进行检查，避免一些敏感词汇。方法 preventFresh() 实现防止恶意刷屏功能。它在会话中保存聊天客户上两次的聊天信息，如果发现三次提交的信息全部相同，则视为恶意刷屏，将利用客户的会话 ID 信息，将其添加到黑名单中，对其发送的信息进行屏蔽。

程序 3-17: chatSessionListener.java

```

package com.util;

import beans.MessageQueue;
import java.util.HashMap;
import javax.servlet.http.HttpSessionListener;
import javax.servlet.http.HttpSessionEvent;

public class chatSessionListener implements HttpSessionListener {
    /**
     * ### Method from HttpSessionListener ###
     *
     * 创建会话时调用。
     */
    public void sessionCreated(HttpSessionEvent evt) {
        // TODO 在此处添加您的代码:
    }

    /**
     * ### Method from HttpSessionListener ###
     *
     * 销毁会话（使其无效）时调用。
     */
    public void sessionDestroyed(HttpSessionEvent evt) {
        // TODO 在此处添加您的代码:
        String sID= evt.getSession().getId();
        String nickname="";
        HashMap users=(HashMap)evt.getSession().getServletContext().getAttribute("USERS");
        if(users!=null) { nickname=(String)users.get(sID);
            users.remove(sID);
        }
    }
}

```



```

        evt.getSession().getServletContext().setAttribute("USERS",users);
    }
    MessageQueue mq=(MessageQueue)evt.getSession().getServletContext().getAttribute
    ("MESSAGE");
    if(mq==null)mq=new MessageQueue();
    mq.setMessage(nickname+"拱手道：青山长在，绿水长流，咱们后会有期，说罢一溜烟跑了");
    evt.getSession().getServletContext().setAttribute("MESSAGE",mq);
}
}

```

程序说明：通过实现 `HttpSessionListener` 接口实现对会话生命周期的监听。为了实现在用户退出聊天室时显示公告信息，在 `sessionDestroyed()` 方法中从 Web 上下文中获取代表聊天内容的 `MessageQueue` 对象，并将公告信息添加到其中，然后将 `MessageQueue` 对象重新保存到 Web 上下文中。

说明：在会话创建时，还不知道用户提交的名称信息，因此无法在 `sessionCreated()` 方法中显示欢迎用户的信息。

程序 3-18: `NewUserListener.java`

```

/*
 *
 * 根据会话中的属性侦听新加入聊天室的用户
 * Created on 2007 年 6 月 25 日, 下午 4:09
 */

package com.util;

import beans.MessageQueue;
import javax.servlet.http.HttpSessionAttributeListener;
import javax.servlet.http.HttpSessionBindingEvent;
public class NewUserListener implements HttpSessionAttributeListener {
    /**
     * ### Method from HttpSessionAttributeListener ###
     *
     * 将属性添加到会话时调用。
     */
    public void attributeAdded(HttpSessionBindingEvent evt) {
        // TODO 在此处添加您的代码:
        String name=evt.getName();
        if(name.endsWith("ID")){
            MessageQueue
mq=(MessageQueue)evt.getSession().getServletContext().getAttribute
("MESSAGE");
            if(mq==null)mq=new MessageQueue();
            mq.setMessage("欢迎"+(String)evt.getValue()+"进入聊天室...");
            evt.getSession().getServletContext().setAttribute("MESSAGE",mq);
        }
    }

    /**
     * ### Method from HttpSessionAttributeListener ###
     *

```

```

    * 从会话中删除属性时调用。
    */
    public void attributeRemoved(HttpSessionBindingEvent evt) {
        // TODO 在此处添加您的代码:
    }

    /**
     * ### Method from HttpSessionAttributeListener ###
     *
     * 当某个属性的值被会话中的其他值替换时调用。
     */
    public void attributeReplaced(HttpSessionBindingEvent evt) {
        // TODO 在此处添加您的代码:
    }
}

```

程序说明: 在 LoginServlet 中, 我们注意到当用户登录成功后, 将当前用户的聊天名称作为属性“ID”加入会话中, 可利用会话属性监听器监听此属性的加入, 然后据此显示欢迎信息。在 attributeAdded() 方法中, 可调用 HttpSessionBindingEvent 的 getName() 方法获取触发事件的属性的名称, 调用 HttpSessionBindingEvent 的 getValue() 方法获取触发事件的属性的值。

程序 3-19: EncodingFilter.java

```

package com.util;

import java.io.*;
import java.net.*;
import java.util.*;
import java.text.*;
import javax.servlet.*;
import javax.servlet.http.*;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class EncodingFilter implements Filter {

    private FilterConfig filterConfig = null;
    private String encoding=null;
    public EncodingFilter() {
    }

    private void doBeforeProcessing(ServletRequest request, ServletResponse response)
    throws IOException, ServletException {
        if (debug) log("EncodingFilter:DoBeforeProcessing");
    }
}

```



```

private void doAfterProcessing(ServletRequest request, ServletResponse response)
throws IOException, ServletException {
    if (debug) log("EncodingFilter:DoAfterProcessing");
}

public void doFilter(ServletRequest request, ServletResponse response,
    FilterChain chain)
    throws IOException, ServletException {
    if (debug) log("EncodingFilter:doFilter()");
    doBeforeProcessing(request, response);

    Throwable problem = null;

    try {
        if(encoding!=null)request.setCharacterEncoding(encoding);
        chain.doFilter(request, response);
    }
    catch(Throwable t) {
        problem = t;
        t.printStackTrace();
    }

    doAfterProcessing(request, response);

    if (problem != null) {
        if (problem instanceof ServletException) throw (ServletException)problem;
        if (problem instanceof IOException) throw (IOException)problem;
        sendProcessingError(problem, response);
    }
}

public FilterConfig getFilterConfig() {
    return (this.filterConfig);
}

public void setFilterConfig(FilterConfig filterConfig) {
    this.filterConfig = filterConfig;
}

public void destroy() {
}

public void init(FilterConfig filterConfig) {
    this.filterConfig = filterConfig;
}

```

```

    if (filterConfig != null) {
        encoding=filterConfig.getInitParameter("encoding");
        if (debug) {
            log("EncodingFilter:Initializing filter");
        }
    }
}

public String toString() {

    if (filterConfig == null) return ("EncodingFilter()");
    StringBuffer sb = new StringBuffer("EncodingFilter()");
    sb.append(filterConfig);
    sb.append(")");
    return (sb.toString());

}

private void sendProcessingError(Throwable t, ServletResponse response) {

    String stackTrace = getStackTrace(t);

    if(stackTrace != null && !stackTrace.equals("")) {
        try {

            response.setContentType("text/html");
            PrintStream ps = new PrintStream(response.getOutputStream());
            PrintWriter pw = new PrintWriter(ps);
            pw.print("<html>\n<head>\n<title>Error</title>\n</head>\n<body>\n");

            pw.print("<h1>The resource did not process correctly</h1>\n<pre>\n");
            pw.print(stackTrace);
            pw.print("</pre></body>\n</html>");
            pw.close();
            ps.close();
            response.getOutputStream().close();
        }

        catch(Exception ex){ }

    }
    else {
        try {
            PrintStream ps = new PrintStream(response.getOutputStream());
            t.printStackTrace(ps);
            ps.close();
            response.getOutputStream().close();
        }

        catch(Exception ex){ }
    }
}

```



```
public static String getStackTrace(Throwable t) {  
    String stackTrace = null;  
  
    try {  
        StringWriter sw = new StringWriter();  
        PrintWriter pw = new PrintWriter(sw);  
        t.printStackTrace(pw);  
        pw.close();  
        sw.close();  
        stackTrace = sw.getBuffer().toString();  
    }  
    catch (Exception ex) {}  
    return stackTrace;  
}  
  
public void log(String msg) {  
    filterConfig.getServletContext().log(msg);  
}  
  
private static final boolean debug = true;  
}
```

程序说明：程序通过过滤器实现对请求提交信息的编码设置，避免显示信息时乱码的产生。

运行结果

程序发布成功后，在浏览器地址栏中输入“http://localhost:8080/MyChat/”，将得到如图 3-9 所示的运行结果。



图 3-9 聊天室登录界面

在文本输入框中输入聊天时使用的昵称，单击【确定】按钮，将得到如图 3-10 所示的运行结果。

在界面的右边可以看到当前在线聊天用户列表。如果在局域网，从多个机器访问此聊天室，可以更好地测试此聊天室的功能。

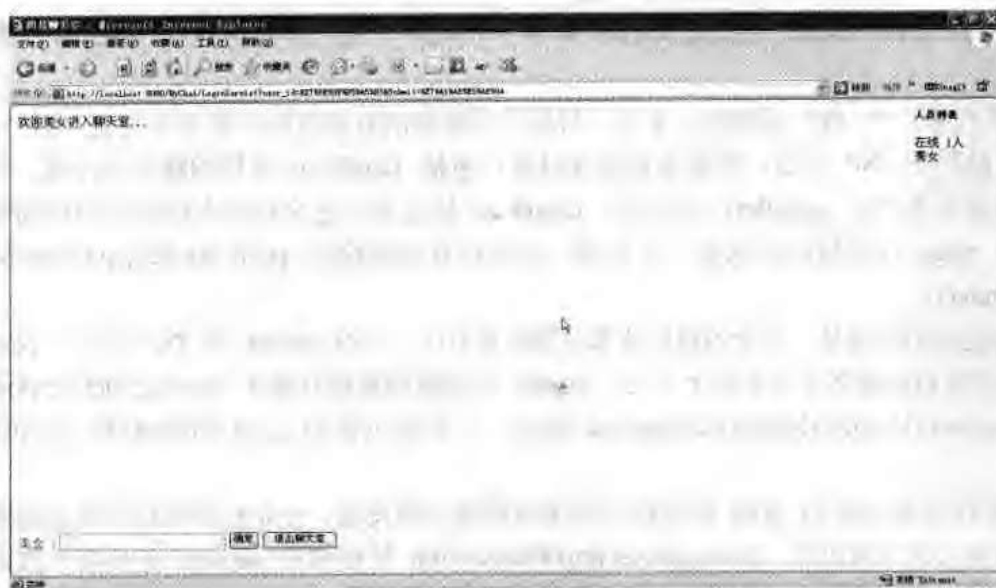


图 3-10 进入聊天室的欢迎界面

讨论与思考

Web 应用程序的状态信息必须存放在特定的环境中,这一特定环境被称为作用域。不同作用域内的数据可支持不同范围的数据访问。在 Java EE 中有 4 种作用域: application (应用程序)、session (会话)、request (请求) 和 page (页面)。

每种 Java EE 作用域都有一个上下文，在这个上下文中可以关联（存储）基本类型的数据和对象引用，以供享有同一上下文的其他组件使用。表 3-1 列出了这 4 种作用域，并说明了它们是否能应用于 Servlet 和 JSP 页面，还给出了对每种作用域的描述。

表 3-1 Java EE 中的作用域

作用域	Servlet	JSP	描述
page	否	是	代表与一个页面相关的对象和属性。一个页面由一个编译好的 Java servlet 类（可以带有任何的 include 指令，但是没有 include 动作）表示。这既包括 servlet 又包括被编译成 servlet 的 JSP 页面
request	是	是	代表与 Web 客户机发出的一个请求相关的对象和属性。一个请求可能跨越多个页面，涉及多个 Web 组件（由于 forward 指令和 include 动作的关系）
session	是	是	代表与用于某个 Web 客户机的一个用户体验相关的对象和属性。一个 Web 会话可以也经常跨越多个客户机请求
application	是	是	代表与整个 Web 应用程序相关的对象和属性。这实质上是跨越整个 Web 应用程序，包括多个页面、请求和会话的一个全局作用域

在上述 4 种作用域中，每种作用域都有一个单独的类，通过这个类可以存储和检索上下文相关的数据。表 3-2 标出了 4 个类，分别对应于 4 种会话作用域。

表 3-2 用于存储有作用域的数据的类

类 名	作 用 域	注 释
javax.servlet.jsp.PageContext	page	特定于 JSP 的一个类型, 代表当前的 JSP 页面。PageContext 实例提供了对所有与一个 JSP 页面相关的名称空间的访问途径
javax.servlet.http.HttpServletRequest	request	代表当前的 HTTP 请求
javax.servlet.http.HttpSession	session	代表当前的 HTTP 会话
javax.servlet.ServletContext	application	这种类型代表整个的运行时 Web 模块

在由 JSP 和 Java Servlet 规范定义的这 4 种作用域中，每一种作用域在 Web 应用程序中都有其明确的用途。

page 提供代表一个 JSP 页面的上下文，与用户所看到的真实页面之间常常具有一对一的映射关系。这种作用域只能用于 JSP 页面，并且也是所有对象，包括 JavaBean 组件的默认作用域。具有 page 作用域的对象通常是那些在 scriptlet、表达式、JavaBean 标记及自定义标记中被访问的局部变量。如果希望获得一个有 page 作用域的对象的一个引用，则可以在该页面的 `javax.servlet.jsp.PageContext` 变量上调用 `getAttribute()`。

request 最适合的环境是：单个的用户请求可能涉及不止一个的 servlet 或 JSP 页面。`request` 是一种能够在一个原子请求内跨越多个页面的上下文。`request` 作用域的数据存储在 `javax.servlet.ServletRequest` 对象中（使用 `javax.servlet.http.HttpServletRequest` object）并通过使用 `getAttribute()` 和 `setAttribute()` 方法来访问。

session 是有状态 Java EE Web 应用程序作用域的核心和灵魂。正是这种作用域使得跨越多个请求的持久用户体验的创建成为可能。`javax.servlet.http.HttpSession` 是存储有 session 作用域的数据的地方，可以通过调用 `getAttribute()` 和 `setAttribute()` 方法来访问。

application 是具有最长运行时间的作用域。这是 Java EE 为全局数据提供的。应用程序数据被一个应用程序模块内的所有 Web 组件所共享。具有应用程序作用域的对象属于 `javax.servlet.ServletContext`，可以通过调用 `getAttribute()` 和 `setAttribute()` 方法来访问。

在使用上述作用域来维护和管理 Web 应用程序的状态信息时，通常需要遵循以下原则。

（1）对于 JSP 数据坚决使用 page 作用域

JSP 页面内所有数据的默认作用域都是 page，它允许用户在为局部变量指定的范围内（类/方法/局部变量作用域）使用这种数据。

说明：需要明确 JSP include 对作用域的影响。page 作用域适用于单个的、经过编译的 Java servlet 类。因为 include 指令是在编译的时候处理的，包括在指令中的任何内容都是在 page 作用域的上行文中操作的。另一方面，include 动作是在运行的时候处理的。如果使用了 include 动作，为了在两个组件之间共享数据，应该使用 request 作用域。

（2）为了在运行的时候在 Web 组件之间共享数据，使用 request 作用域

如果使用一个 forward 动作或者 include 动作来在两个或更多组件之间共享一个请求，那么通过将数据的作用域设为 request 作用域便可以在这些组件之间共享该数据。

（3）为了提供有状态用户体验，使用 session 作用域

无论要构建的是在线商店、电子邮件管理站点、个人化信息门户，还是财务管理应用，session 作用域都是对用户采取从请求到请求的跟踪或者为用户提供无缝的、持久的环境的最佳选择。

（4）将 application 作用域专用于全局数据

应用程序对象是一些静态的对象，为应用程序内的所有对象所共享。对 application 作用域的使用应该保留给真正需要在组件之间共享或者跨用户会话的数据。典型的例子有：缓存的数据访问对象（Data Access Object, DAO），Java 名称和目录接口（Java Name and Directory Interface, JNDI）引用或者任何类型的公共工厂或者其他需要使用 Singleton 模式的组件。

知识点索引

HttpRequest; HttpSession; ServletContext; 过滤器; 监听器。

例程 3-3：网站计数器

目的

- (1) 应用程序状态在外部资源文件中的持久化存储；
- (2) 利用监听器监视应用程序状态变化；
- (3) 维护应用程序状态时的线程安全问题。

问题

如何创建一个网站计数器组件来记录客户访问网站的次数？并满足以下要求：

- (1) 统计应用自部署以来的所有用户访问次数；
- (2) 对于用户在一次会话中的访问只记录一次，以保证数据的真实性；
- (3) 统计在线用户数量信息。

解决方案

由于网站计数器要记录应用自部署以来的所有用户访问次数，因此，必须将用户访问信息实现持久化。由于用户访问信息比较简单，因此，可以将信息持久化存储到外部的资源文件 `count.txt` 而不是数据库中。

利用 Web 上下文监听器控制历史计数信息的读取和写入。当 Web 应用上下文创建时，将历史计数信息从外部资源文件读取到内存。当 Web 应用上下文关闭时，则将历史计数信息持久化存储到外部资源文件。

利用会话监听器监听在线用户数量变化。会话创建事件，每创建一个新的会话，则代表产生一次新的用户访问。每一个会话销毁事件，则代表一位用户离线。这种方式也避免了用户重复刷新导致的重复计数。

实现步骤

- (1) 创建辅助工具类 `CounterFile` 操作资源文件 `count.txt`；
- (2) 创建 Web 上下文监听器调用辅助工具类 `CounterFile` 控制历史计数信息的操作；
- (3) 创建会话监听器维护在线用户信息；
- (4) 创建 JSP 页面显示历史计数信息和在线用户信息。

示例代码

本例程的所有示例代码均在 `netbeans` 的工程 `Counter` 内。

程序 3-20: `CounterFile.java`

```
package com.example;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
```



```

import java.io.IOException;
import java.io.PrintWriter;
//用来操作记录访问次数的文件
public class CounterFile {
    private BufferedReader file; //BufferedReader 对象，用于读取文件数据
    /** Creates a new instance of CounterFile */
    public CounterFile() {
    }
    //ReadFile 方法用来读取文件 filePath 中的数据，并返回这个数据
    public String ReadFile(String filePath) throws FileNotFoundException {
        String currentRecord = null; //保存文本的变量
        //创建新的 BufferedReader 对象
        file = new BufferedReader(new FileReader(filePath));
        String returnStr = null;
        try {
            //读取一行数据并保存到 currentRecord 变量中
            currentRecord = file.readLine();
        } catch (IOException e) { //错误处理
            System.out.println("读取数据错误。");
        }
        if (currentRecord == null)
            //如果文件为空
            returnStr = "没有任何记录";
        else { //文件不为空
            returnStr = currentRecord;
        }
        //返回读取文件的数据
        return returnStr;
    }
    //ReadFile 方法用来将数据 counter+1 后写入到文本文件 filePath 中
    //以实现计数增长的功能
    public synchronized void WriteFile(String filePath, String counter) throws
    FileNotFoundException {

        int Writestr = 0;
        Writestr = Integer.parseInt(counter);
        try {

            //创建 PrintWriter 对象，用于写入数据到文件中
            PrintWriter pw = new PrintWriter(new FileOutputStream(filePath));
            //用文本格式打印整数 Writestr
            pw.println(Writestr);
            //清除 PrintWriter 对象
            pw.close();
        } catch (IOException e) {
            //错误处理
            System.out.println("写入文件错误" + e.getMessage());
        }
    }
}

```

程序说明：程序用来实现对资源文件 count.txt 的读写操作。其中 ReadFile(String filePath) 方法负责将信息从资源文件读取到内存，WriteFile(String filePath,String counter)方法负责将计数信息写回到资源文件。为了避免写文件时发生线程安全问题，这里将 WriteFile 方法用 synchronized 加以保护。

程序 3-21: CounterListener.java

```
package com.example;

import javax.servlet.ServletContextListener;
import javax.servlet.ServletContextEvent;

public class CounterListener implements ServletContextListener {

    String path="";

    public void contextInitialized(ServletContextEvent evt) {

        CounterFile f=new CounterFile();
        String name=evt.getServletContext().getInitParameter("CounterPath");
        path=evt.getServletContext().getRealPath(name);
        try{
            String temp=f.ReadFile(path);
            System.out.println("this is my reading.....");
            System.out.println(temp);
            evt.getServletContext().setAttribute("Counter",temp); //将计数器的值放入应用上下文
        }catch(Exception e){
            System.out.println(e.toString());
        }

    }

    public void contextDestroyed(ServletContextEvent evt) {
        try{
            String current= (String)evt.getServletContext().getAttribute("Counter");

            CounterFile f=new CounterFile();
            f.WriteFile(path,current);
        }catch(Exception e){
            System.out.println(e.toString());
        }

    }

}
```

程序说明：程序调用辅助工具类 CounterFile 来操作资源文件。在 contextInitialized(ServletContextEvent evt)方法中将历史计数信息从外部读入到内存，contextDestroyed(ServletContextEvent evt)方法中将历史计数信息持久化储存到外部文件。这样，在程序运行期间，不管应用的访问次数多么频繁，所有计数操作只是操作内存中的变量，应用程序的性能将不会因为频繁的 IO 操作而下降。

程序 3-22: SessionListener.java

```
package com.example;
```



```

import javax.servlet.http.HttpSessionListener;
import javax.servlet.http.HttpSessionEvent;
public class SessionListener implements HttpSessionListener {

    public void sessionCreated(HttpSessionEvent evt) {
        // 修改在线人数
        String current= (String)evt.getSession().getServletContext().getAttribute("online");
        if(current==null)current="0";
        int c=Integer.parseInt(current);
        c++;
        current=String.valueOf(c);
        evt.getSession().getServletContext().setAttribute("online",current);
        //修改历史人数
        String his= (String)evt.getSession().getServletContext().getAttribute("Counter");
        if(his==null)his="0";
        int total=Integer.parseInt(his)+1;
        his=String.valueOf(total);
        evt.getSession().getServletContext().setAttribute("Counter",his);
    }

    public void sessionDestroyed(HttpSessionEvent evt) {
        // TODO 在此处添加您的代码:
        // 修改在线人数
        String current= (String)evt.getSession().getServletContext().getAttribute("online");
        if(current==null)current="0";
        int c=Integer.parseInt(current);
        c--;
        current=String.valueOf(c);
        evt.getSession().getServletContext().setAttribute("online",current);
    }
}

```

程序说明：程序通过监听会话事件来维护在线人数和历史访问次数信息。在 `sessionCreated(HttpSessionEvent evt)` 方法中，从 Web 应用上下文中获取历史计数信息和在线人数信息，并分别增加 1。在 `sessionDestroyed(HttpSessionEvent evt)` 方法中，从 Web 应用上下文中获取在线人数信息，并减 1。

程序 3-23: counter.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%
String history =(String)application.getAttribute("Counter");
if( history==null) history="0";
int last=Integer.parseInt( history);
String temp =(String)application.getAttribute("online");
if(temp==null)temp="0";
int online=Integer.parseInt(temp);
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

```

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title></title>
  </head>
  <body>

    <p align="center">欢迎! 您是第<font color="red"><%=last%></font>位访问者</p>
    <p align="center">当前在线用户<font color="red"><%=online%></font>人</p>
  </body>
</html>
```

程序说明：用来从 Web 应用上下文属性中获取历史访问次数和在线人数信息，并通过 JSP 脚本输出到页面。

运行结果

程序发布成功后，在浏览器地址栏输入“<http://localhost:8080/Counter/counter.jsp>”，将得到如图 3-11 所示的运行结果。

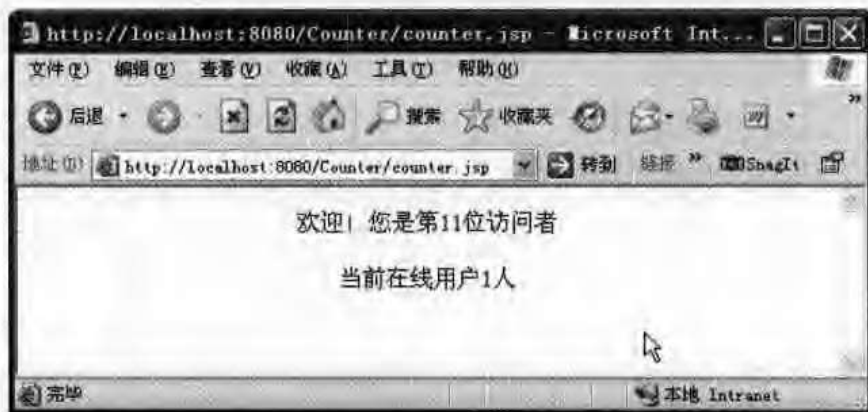


图 3-11 网站计数器

关闭服务器，然后重新启动，可以看到页面的访问信息被成功地持久化存储。

讨论与思考

Web 应用中的事件标志着 Web 应用的状态和属性的变化。对于 Web 应用中的事件的监听，在许多应用开发中具有十分重要的意义。如果用户希望自己的类能够监听应用的启动和停止事件，则必须实现 `ServletContextListener` 接口；如果希望类能够监听 Session 的创建和失效事件，则必须实现 `HttpSessionListener` 接口。

知识点索引

Web 上下文监听器；会话监听器；持久化存储。

本章小结

管理和维护应用程序的状态信息是 Java EE 编程的最基本,也是最重要的任务之一,这是由目前 Web 应用程序的基本模式决定的。HTTP 协议的无状态特性导致程序开发人员必须肩负起管理和维护应用程序的状态信息的重任。

管理和维护应用程序状态的首要任务是实现会话跟踪。例程 3-1 中展示了各种会话跟踪技术,Java EE 规范提供的 HttpSession 无疑是一种最便捷的方法。Web 应用程序的状态数据有四种不同的作用域:page, request, session 和 application。各种不同的作用域适合保存不同的状态信息,开发人员要根据实际情况,灵活运用。例程 3-2 展示了如何应用不同的作用域来保存应用的状态信息。对于状态信息的变化,Java EE 最新的规范提供了监听器,可以帮助开发人员及时发现并处理这种变化。



第4章 访问企业信息资源

团结就是力量

——谚语

信息的价值在于共享，只有将分散的信息集成起来，才能发挥更大的威力，创造更大的经济效益和社会效益，这也是信息化的本质。打破“烟囱式”的信息系统建设模式，建设规范的集成的一体化信息平台，将各分散的信息系统连接起来，已经成为当前业界的共识。

作为开放的企业分布式应用的开发标准，Java EE 为与各种企业信息系统连接提供了技术支撑。本章将通过多个示例演示如何在 Java EE 框架下连接包括数据库服务器、邮件服务器在内的各种企业信息系统。

例程 4-1：发送接收 E-mail

目的

演示在 Web 应用实现与 E-mail 系统的交互。

问题

如何为 Web 应用系统添加收发 E-mail 的功能特性？

解决方案

Java EE 规范中提出了 Java Mail，可以支持 Web 应用与电子 Java Mail 邮件系统进行交互。利用 Java Mail，实现为 Web 应用系统添加收发 E-mail 的功能特性。

知识链接

Java Mail API 是一个用于阅读、编写和发送电子消息的可选包(标准扩展)，可以用来建立标准的电子邮件客户端程序，它支持各种因特网邮件协议，包括 SMTP，POP，IMAP，MIME 等。

Java EE 通过 Java Mail API 为 Java 程序开发者提供了一个访问邮件服务器的通用接口。整个 Java Mail 体系可以分为三层：抽象层、Internet 邮件实现层和协议实现层。如图 4-1 所示，Java Mail API 包括抽象层和 Internet 邮件实现层。

Java Mail 抽象层：该层定义了用于邮件处理功能的抽象类、接口和抽象方法，所有的邮件系统都支持这些功能，它独立于供应商和协议消息。抽象层位于 Java Mail 顶级包（即 javax.mail）内。

Internet 邮件实现层：该层实现了部分抽象层元素，它遵循 Internet 标准——RFC822 和 MIME。Internet 邮件实现层所定义的和接口大多位于 javax.mail.internet 包内。

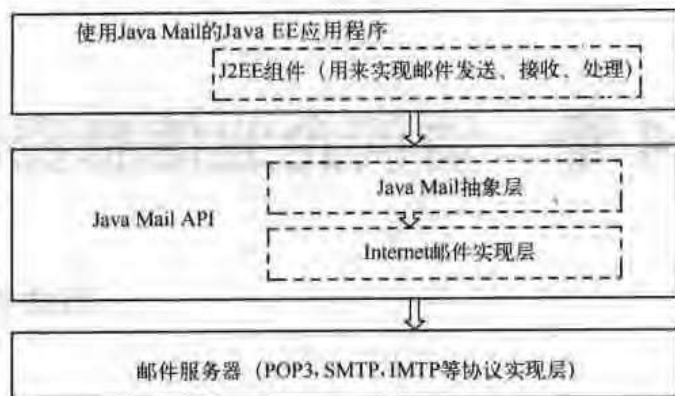


图 4-1 Java Mail 体系结构

协议实现层：该层由服务提供商实现对特定协议的支持，如 SMTP，POP，IMAP 和 NNTP。Sun 公司为开发者提供了很完备的一组协议的支持。

Java Mail 客户机和邮件服务供应商通过 Java Mail API 进行交互。这种体系确保无论使用哪种协议，由哪一家厂商提供对该协议的实现，Java Mail 客户机都可以使用同样的 Java Mail API 收发邮件。

运行准备

Netbeans 内嵌的 Sun Java Application Server 已经包含了 Java Mail 编程需要的 Java Mail 扩展包和 JavaBeans Activation Framework (JAF)，因此不再需要添加额外的 Jar 到工程的类路径下，如果使用其他应用服务器，如 Tomcat，则需要将上述两个文件下载后添加到工程的类路径下。

实现步骤

- (1) 创建 javax.mail.Authenticator 的继承类，作为邮件发送的辅助类；
- (2) 创建 Servlet SendMail 实现邮件发送功能；
- (3) 创建静态页面 MailLogon.html 来登录邮件服务器；
- (4) 创建 Servlet ReceiveSimpleMail 实现邮件接收功能；
- (5) 创建 JSP 页面 mailcontent 显示邮箱信息。

示例代码

本例程的所有示例代码均在 netbeans 的工程 mail 内。

程序 4-1: Auth.java

```

package com.mail;
import javax.mail.Authenticator;
import javax.mail.PasswordAuthentication;
public class Auth extends Authenticator {
    String username="";
    String password="";
    public Auth(String username,String password){
        this.username=username;
        this.password=password;
    }
    public PasswordAuthentication getPasswordAuthentication(){
        return new PasswordAuthentication( username, password);
    }
}
  
```

程序说明：大部分商业邮件服务器在发送邮件都需要实现认证功能。程序继承了 `javax.mail.Authentication`，作为辅助类，它主要用来实现登录邮件服务器时的认证。它包含两个属性：`username` 和 `password`，用来存储认证时所需的用户名和密码信息。它还覆盖 `Authenticator` 类的 `getPasswordAuthentication()` 方法。`getPasswordAuthentication()` 方法必须声明为 `protected` 类型，返回一个 `PasswordAuthentication` 类型的对象，此 `PasswordAuthentication` 对象包含了通过 SMTP 服务器身份验证所需的用户名和密码，供认证时 Session 调用。

程序 4-2: SendMail.java

```
package com.mail;

import java.io.IOException;
import java.io.PrintWriter;
import javax.mail.Message.RecipientType;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.mail.*;
import java.util.*;
import javax.mail.internet.*;

public class SendMail extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");

        PrintWriter out = response.getWriter();
        try{
            String host = "smtp.163.com" ;
            //创建一个属性对象
            Properties props = new Properties();
            //指定发送邮件服务器，使用 SMTP 协议
            props.put("mail.smtp.host", host);
            props.put("mail.smtp.auth", "true");

            Auth au=new Auth("haoyulongsd@163.com","ibmt20@sd");
            Session sendMailSession =Session.getInstance(props, au);

            // sendMailSess
            Message newMessage = new MimeMessage(sendMailSession);
            newMessage.setFrom(new InternetAddress(request.getParameter("from")));
            newMessage.setRecipient(RecipientType.TO, new InternetAddress
                ( request.getParameter ("to")));
```



```

        newMessage.setSubject(request.getParameter("subject"));
        newMessage.setSentDate(new Date());
        newMessage.setText(request.getParameter("text"));
        Transport.send(newMessage);
        ///显示发送成功的提示
        out.println(" the Email send sucess!");
    } catch (Exception m)
    {
        out.println(m.toString());
    }
    out.close();
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}
}

```

程序说明：首先通过语句 `props.put("mail.smtp.auth","true")` 设置邮件会话需要认证，然后利用用户名和密码信息生成认证对象，最后根据属性对象和认证对象调用会话对象的 `getInstance(Properties prop, Authenticator auth)` 来获取连接邮件服务器的 Session 对象实例。如果认证成功，则获得 Session 对象实例，如果认证不成功，则抛出意外。

程序 4-3: MailLogon.html

```

<html lang='zh'>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
    <title> JavaMail - 登录邮件服务器</title>
  </head>
  <body>
    <table border="0" cellspacing="0" cellpadding="0">
      <form method="post" action="ReceiveSimpleMail">
        <tr>
          <td>用户名称: </td>
          <td><input type="text" name="user" size="80"></td>
        </tr>
        <tr>
          <td>用户密码: </td>
          <td><input type="password" name="password" size="80"></td>
        </tr>
        <tr>
          <td>邮局协议: </td>
          <td><input type="text" name="protocol" size="80"></td>
        </tr>
      </form>
    </table>
  </body>
</html>

```

```

</tr>
<td colspan="2" align="center"><input type="submit" value="接收邮件"></td>
</tr>
</form>
</table>
</body>
</html>

```

程序说明：页面主要用来收集登录邮件服务器所需的各种信息，并将其发送到后端 Servlet 进行处理。

程序 4-4: ReceiveSimpleMail.java

```

package com.mail;

import java.io.*;
import java.util.Properties;
import javax.mail.Folder;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Store;
import javax.mail.internet.MimeMessage;

import javax.servlet.*;
import javax.servlet.http.*;

public class ReceiveSimpleMail extends HttpServlet {

    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        doPost(request, response);
    }

    protected void doPost(HttpServletRequest request,
        HttpServletResponse response) throws ServletException, IOException {
        String host = "pop3.163.com" ;
        String user=request.getParameter("user");
        String password=request.getParameter("password");
        String protocol=request.getParameter("protocol");
        String url=null;
        String root=null;
        try{
            Properties props = System.getProperties();
            Session sess= Session.getDefaultInstance(props ,null);
            sess.setDebug(true);
            Store store = null;
            store = sess.getStore(protocol);
            store.connect(host, user, password);
            Folder folder = store.getFolder("INBOX");
            folder.open(Folder.READ_ONLY);
            Message message[] = folder.getMessages();
            int size=message.length;
            if(size==0){

```



```

        RequestDispatcher rd = request.getRequestDispatcher("/emptyMailBox.jsp");
        rd.forward(request, response);
    }
    MimeMessage mm = (MimeMessage) message[0];
    request.setAttribute("mail", mm);
    RequestDispatcher rd = request.getRequestDispatcher("/mailcontent.jsp");
    rd.forward(request, response);
} catch (MessagingException e) {
    System.out.println(e.toString());
}
}

public String getServletInfo() {
    return "Short description";
}
}

```

程序说明：首先创建 Session，然后根据 Session 创建 Store 对象，Store 对象映射了邮箱的结构。在调用 Session 对象的 `getStore()` 方法时需要指定所使用的协议，如果客户邮箱是 POP3 邮箱，那么应该使用 `getStore("pop3")`。如果客户邮箱是 IMAP4 类型的那么应该使用 `getStore("imap")`。调用 Store 对象的 `getFolder()` 获取邮箱中指定文件夹中的内容。参数为“INBOX”代表为收件箱。调用 `getFolder()` 方法以后需要打开文件夹然后才能够从文件夹中读取邮件。打开文件夹以后使用 Folder 对象的 `getMessages()` 方法就可以返回一个 Message 数组，里面包含了当前文件夹中所有邮件的列表。将最近的一封邮件信息添加到请求中，交给后面的页面显示。

程序 4-5: mailcontent.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ page import="javax.mail.*" %>
<%@ page import="javax.mail.internet.*" %>
<%
MimeMessage mm = (MimeMessage) request.getAttribute("mail");
out.println("主题: " + mm.getSubject() + "<br>");
out.println("编码: " + mm.getEncoding() + "<br>");
out.println(" ID : " + mm.getMessageID() + "<br>");
out.println("发送日期: " + mm.getSentDate() + "<br>");
out.println("接收日期: " + mm.getReceivedDate() + "<br>");
out.println("大小: " + mm.getSize() + "byte<br>");
out.println("标志: " + "");
mm.setFlag(Flags.Flag.RECENT, true);
Flags flag = mm.getFlags();
if (flag.contains(Flags.Flag.ANSWERED))
{
    out.println("已回复");
}
else if (flag.contains(Flags.Flag.DELETED))
{
    out.println("已删除");
}
else if (flag.contains(Flags.Flag.DRAFT))

```

```
{
    out.println("草稿");
}
else if (flag.contains(Flags.Flag.FLAGGED))
{
    out.println("标记");
}
else if (flag.contains(Flags.Flag.RECENT))
{
    out.println("最近");
}
else if (flag.contains(Flags.Flag.SEEN))
{
    out.println("已阅");
}
else if (flag.contains(Flags.Flag.USER))
{
    out.println("USER");
}
else
{
    out.println("无标记");
}
}
out.println("<br>");
out.println("Folder "+mm.getFolder().getURLName().toString()+"<br>");
out.println("From "+"");
Address []addr=mm.getFrom();
for(int i=0;i<addr.length;i++)
{
    out.println(""+addr[i].toString()+" &nbsp;");
}
out.println("<br>");
out.println("To & CC & BCC : "+"");
Address []addrTo=mm.getAllRecipients();
for(int i=0;i<addrTo.length;i++)
{
    out.println(""+addrTo[i].toString()+" &nbsp;");
}
out.println("<br>");
out.println("Reply To "+"");
Address []addrReply=mm.getReplyTo();
for(int i=0;i<addrTo.length;i++)
{
    out.println(""+addrReply[i].toString()+" &nbsp;");
}
out.println("<br>");
out.println("Content Language "+mm.getContentLanguage()+"<br>");
out.println("Content "+mm.getContent()+"<br>");
%>
```


程序说明：首先调用 `request.getAttribute("mail")` 获取请求中传递来的 `MimeMessage` 对象，然后调用 `MimeMessage` 对象各方法获取邮件的主题、大小、发送时间、状态标记、文件夹、内容等信息。

运行结果

首先演示如何发送 E-mail。程序部署成功后，在浏览器的地址栏输入 “`http://localhost:8080/Mail/`”，得到的运行结果如图 4-2 所示。



图 4-2 发送邮件

在对应的输入框输入相关信息，单击【发送】，则邮件成功发出，将得到如图 4-3 所示的运行结果。



图 4-3 发送邮件成功提示

可以利用 Foxmail、Outlook 等工具登录目标邮箱检查邮件是否成功发送。

下面演示接收邮件。

在浏览器的地址栏输入 “`http://localhost:8080/Mail/MailLogon.html`”，得到的运行结果如图 4-4 所示。



图 4-4 登录邮件服务器

输入登录邮件服务器的用户名称、密码及邮局协议名称,单击【接收邮件】,则根据输入的信息登录邮件服务器并获取邮件信息,得到的运行结果如图4-5所示,可以看到邮件信息已经成功获取。

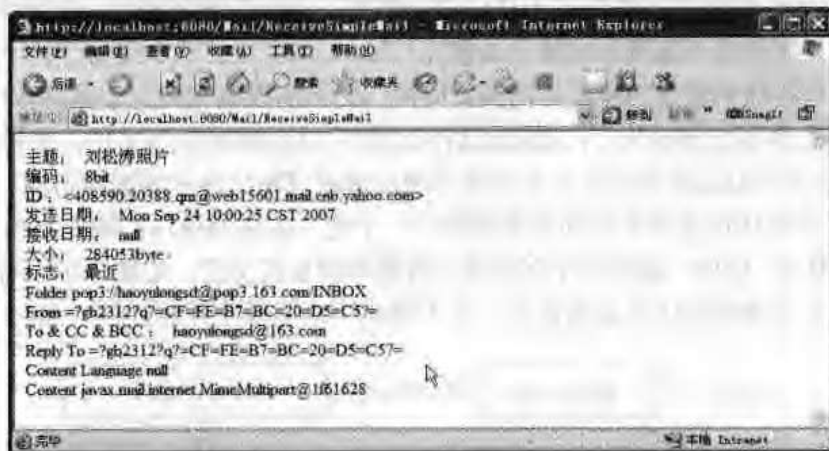


图 4-5 显示邮件信息

讨论与分析

目前开发的许多 Web 应用程序经常需要具有发送、接收和处理电子邮件的功能。Java Mail API 是一个用于阅读、编写和发送电子消息的可选包(标准扩展),可以用来建立标准的电子邮件客户端程序,它支持各种因特网邮件协议,包括 SMTP, POP, IMAP, MIME 等。

知识点索引

JavaMail; Servlet; JSP。

例程 4-2: 访问数据库

目的

演示如何利用 Java EE 技术访问数据库中的信息。

问题

如何访问存放在数据库 test 的表 Users 中的用户信息?

解决方案 1: 直接使用 JDBC 驱动访问数据库

知识链接

Java EE 提供了一个标准接口 JDBC (Java DataBase Connection, Java 数据库连接)来进行数据库访问操作。JDBC 为多种关系数据库提供了统一的访问接口。JDK 1.4 以上版本中已包含了 JDBC。

JDBC 为多种关系数据库提供了统一访问方式,作为特定厂商数据库访问 API 的一种高级抽象,它

主要包含一些通用的接口类（通过查看 JDBC 包中的内容就可以证实这一点）。那么真正的数据库访问操作是在哪里实现的呢？实际上，真正的数据库访问操作实现是由各自数据库厂商提供的。通常把厂商提供的特定于数据库的访问 API 称为数据库 JDBC 驱动程序。JDBC 通过提供一个抽象的数据库接口，使得程序开发人员在编程时可以不用绑定在特定数据库厂商的 API 上，大大增加了应用程序的可移植性。在实际运行过程中，程序代码通过 JDBC 访问数据库时，仍旧需要调用特定于数据库的访问 API。

在基于 JDBC 的数据库访问模式下，数据库访问过程可以清晰地分为 3 层，如图 4-6 所示。最上层为应用层，Java EE 程序开发成员在程序开发过程中通过调用 JDBC 进行数据库访问；中间层为 JDBC 接口层，它为 Java EE 程序访问各种不同的数据库提供一个统一的访问接口；最底层为 JDBC 驱动层，它由特定数据库厂商提供的 JDBC 驱动程序来实现与数据库的真正交互。由图中可以看出，JDBC 将程序员开发的应用程序与具体的数据库产品隔离开，大大简化了应用程序开发过程，提高了程序的可移植性。

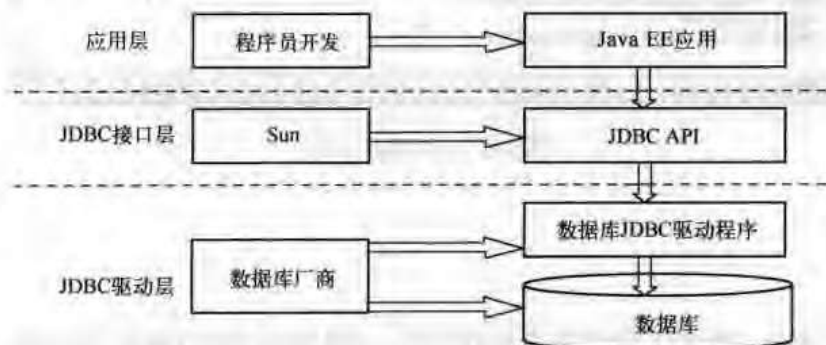


图 4-6 基于 JDBC 的数据库访问体系

实现步骤

- (1) 注册数据库驱动程序；
- (2) 根据 JDBC URL 地址，调用 DriverManager 对象的 getConnection() 来获取到数据库的连接对象 java.sql.Connection；
- (3) 利用连接对象创建代表要执行的 SQL 语句的 Statement 对象；
- (4) 执行 Statement 对象；
- (5) 对执行 Statement 对象后返回的 ResultSet 对象进行分析处理，获取数据库中的信息；
- (6) 关闭 ResultSet 对象；
- (7) 关闭 Statement 对象；
- (8) 关闭 Connection 对象。

示例代码

本例程的所有示例代码均在 netbeans 的工程 news 内。

程序 4-6: Jdbc.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%
    java.sql.Connection conn;
    java.lang.String strConn;
    java.sql.Statement sqlStmt; //语句对象
    java.sql.ResultSet sqlRst; //结果集对象
```

```

try(
    Class.forName("com.mysql.jdbc.Driver").newInstance();
    conn=java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/test",
        "root","javaee");
    sqlStmt=conn.createStatement(java.sql.ResultSet.TYPE_SCROLL_INSENSITIVE,java.
        sql.ResultSet.CONCUR_READ_ONLY);
    String sqlQuery="select id,name,address from users";
    sqlRst=sqlStmt.executeQuery(sqlQuery);
    if(sqlRst!=null){
%>
<center>用户信息表</center>
<table border="1" width="70%" bordercolorlight="#CC99FF" cellpadding="2"
bordercolordark="#FFFFFF" cellspacing="0">
    <tr>
        <td align="center">&nbsp;ID</td>
        <td align="center">姓名</td>
        <td align="center">地址电话</td>
    </tr>
    <% while (sqlRst.next()) { //取得下一条记录%>
    <tr><!--显示记录-->
        <td align="center"><%=sqlRst.getString("id")%></td>
        <td align="center"><%=sqlRst.getString("name")%></td>
        <td align="center"><%=sqlRst.getString("address")%></td>
    </tr>
    <% } %>
</table>

<%
    }

//关闭结果集对象
    sqlRst.close();
//关闭语句对象
    sqlStmt.close();
//关闭数据库连接
    conn.close();
} catch (java.sql.SQLException e){
    out.println(e.toString());
}

%>
</body>
</html>

```

运行准备

打开 MySQL 数据库，首先建立一个存储用户信息的专用数据库 test。系统中共有 1 个表格：用户信息表 users，用来存储用户的登录信息。表格的结构信息如表 4-1 所示。

表 4-1 用户信息表 users 的结构信息

字段名	字段类型	备注	字段说明
id	Varchar (6)	主键, 系统自增字段	用户 ID
name	Varchar (32)		用户名称
address	Varchar (200)		用户住址

提示: 可以根据以上信息在数据库中建立相应表格。也可通过脚本文件 test.sql 来还原数据库。MySQL 有很多客户端软件 (如 EngInSite MySQL Client) 可以帮助开发人员还原数据库。

运行结果

应用部署成功后, 在浏览器的地址栏输入 “http://localhost:8080/News/Jdbc.jsp”, 将得到如图 4-7 所示的运行结果。



图 4-7 直接利用 JDBC 访问数据库

解决方案 2: 利用 JDBC-ODBC 桥访问数据库

知识链接

Java 应用程序通过 JDBC API 与数据库连接, 而实际的动作则是由 JDBC 驱动程序管理器 (JDBC Driver Manager) 通过 JDBC 驱动程序与数据库系统进行连接。尽管 JDBC 已经成为业界的一种标准, 但并不是所有的数据库都支持 JDBC, 特别是一些老的数据库如 Access。除了 JDBC 外, 还有另外一个访问关系数据库的标准接口 ODBC (Open DataBase Connectivity), 即开放式的接口, 对于不同的数据库它提供了一套统一的 API: 可以使应用程序通过 API 访问任何提供了 ODBC 驱动程序的数据库。

JDBC-ODBC 桥是一种 JDBC 驱动程序, 它通过将 JDBC 操作转换为 ODBC 操作来实现的。因此, JDBC-ODBC 桥为访问不支持 JDBC 驱动的应用程序提供了一种转换途径。另外, 利用 JDBC-ODBC 桥可以使程序开发人员不需要学习更多的知识就可以编写 JDBC 应用程序, 并能够充分利用现有的 ODBC 数据源。

运行准备

步骤 1: 创建 Access 数据库

创建 Access 数据库 MyAccess, 它包含用户信息表 Users。表 Users 的结构与前面完全一致。

步骤 2: 为 Access 创建 ODBC 数据源

具体操作如下: 单击【开始】→【设置】→【控制面板】→【管理工具】→【数据源 (ODBC)】, 弹出【ODBC 数据源管理】对话框, 如图 4-8 所示。

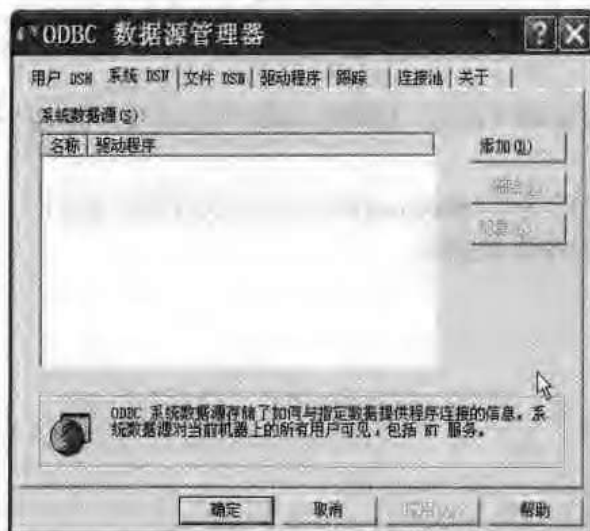


图 4-8 添加 ODBC 数据源

选中【系统 DSN】标签，单击右侧的【添加】按钮，弹出【创建新数据源】对话框，如图 4-9 所示。



图 4-9 为 ODBC 数据源添加驱动程序

选择驱动程序【Driver do Microsoft Access (*.mdb)】，单击【完成】按钮，则进入数据库选择页面，如图 4-10 所示。



图 4-10 为数据源选择数据库

在【数据源名】输入框中输入 ODBC 数据源的名称“jdbcodbc”，单击【选择】按钮，选中要访问的 Access 数据库，最后单击【确定】按钮完成数据源的创建。

实现步骤

- (1) 加载 JDBC—ODBC 驱动（说明：标准的 JDK 中已经包含 JDBC—ODBC 驱动，因此，不需要额外的包加入）；
- (2) 调用 DriverManager 的 getConnection(url) 方法获取数据库连接；
- (3) 利用数据库连接对象操作数据库；
- (4) 关闭数据库连接。

示例代码

程序 4-7: testjdbcodbc.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ page import="java.sql.*" %>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>用 JDBC—ODBC 访问数据库</title>
    </head>
    <body>

        <%
            try{
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                String url = "jdbc:odbc:jdbcodbc";
                Connection conn = DriverManager.getConnection(url);
                //如果设置用户名 user 和密码 pwd, 则代码如下:
                //Connection conn = DriverManager.getConnection(url,"user","pwd");
                Statement stmt = conn.createStatement();
                //下面的 select 语句中需要注意, 一定要在 test.mdb 中有 Table1 这个表, 也可以改成实际存在的表名
                ResultSet rs = stmt.executeQuery("SELECT id,name,address FROM Users");
                if(rs!=null){%>
                    <center>用户信息表</center>
                    <table border="1" width="70%" bordercolorlight="#CC99FF" cellpadding="2"
                        bordercolordark="#FFFFFF" cellspacing="0">
                        <tr>
                            <td align="center">&nbsp;ID</td>
                            <td align="center">姓名</td>
                            <td align="center">地址电话</td>
                        </tr>
                        <% while (rs.next()) { //取得下一条记录%>
                            <tr><!--显示记录-->
                                <td align="center"><%=rs.getString("id") %></td>
                                <td align="center"><%=rs.getString("name") %></td>
                                <td align="center"><%=rs.getString("address") %></td>
                            </tr>
                        <% } %>
                    </table>
```

```

<%
    }

    //关闭结果集对象
    rs.close();
    //关闭语句对象
    stmt.close();
    //关闭数据库连接
    conn.close();
} catch (java.sql.SQLException e){
    out.println(e.toString());
}

%>
</body>
</html>

```

运行结果

应用部署成功后,在浏览器的地址栏输入 <http://localhost:9080/News/testjdbcodbc.jsp>,将得到如图 4-11 所示的运行结果。



图 4-11 利用 JDBC-ODBC 数据源访问数据库

解决方案 3: 利用数据源和连接池技术访问数据库

知识链接

1. 数据源

数据源是在 JDBC 2.0 中引入的一个概念。在 JDBC 2.0 扩展包中定义了 `javax.sql.DataSource` 接口来描述数据源的概念。在数据源中存储了所有建立数据库连接的信息。就像通过指定文件名可以在文件系统中找到文件一样,通过提供正确的数据源名称,应用程序可以找到相应的数据库连接。`javax.sql.DataSource` 接口就定义了如何实现数据源。

如果用户希望建立一个数据库连接,通过查询在 JNDI 服务中的数据源,可以从数据源中获取相应的数据库连接。这样程序开发人员就只需要获取一个逻辑名称,而不是数据库登录的具体细节,这样代码的移植能力就更强。

2. 数据库连接池

传统的数据库连接方式（指通过 `DriverManager` 和基本的数据源进行连接）中，一个数据库连接对象均对应一个物理数据库连接，数据库连接的建立和关闭对系统而言是耗费系统资源的操作，在多层结构的应用程序环境中这种耗费资源的动作对系统的性能影响尤为明显。

在多层结构的应用程序中通过连接池（`Connection Pooling`）技术可以使系统的性能明显得到提高，连接池意味着当应用程序需要调用一个数据库连接时，数据库相关的接口通过返回一个重用的数据库连接来代替创建一个新的数据库连接。通过这种方式，应用程序可以减少对数据库的连接操作，尤其在多层环境中多个客户端可以通过共享少量的物理数据库连接来满足系统需求。通过连接池技术，Java 应用程序不仅可以提高系统性能，同时也为系统提高了可测量性。

数据库连接池是运行在后台的，因此应用程序的编码没有任何的影响。此种状况存在的前提是，应用程序必须通过 `DataSource` 对象（一个实现 `javax.sql.DataSource` 接口的实例）的方式代替原有通过 `DriverManager` 类来获得数据库连接的方式。一个实现 `javax.sql.DataSource` 接口的类既可以支持也可以不支持数据库连接池，但是两者获得数据库连接的代码基本是相同的。

在 JDBC 3.0 规范中，数据库连接池的实现可以分为 JDBC Driver 级和 Application Server 级。在 JDBC Driver 级的实现中，任何相关的工作均由特定数据库厂商的 JDBC Driver 的开发人员来具体实现，即 JDBC Driver 既需要提供对数据库连接池的支持，同时也必须对数据库连接池进行具体实现。而在 Application Server 级的数据库连接池的实现中，特定数据库厂商的 JDBC Driver 开发人员和 Application Server 开发人员共同实现数据库连接池的实现。

实现步骤

（1）注册 MySQL 数据库驱动程序。具体操作如下：在【运行环境】视图下，选中【数据库】路径下的【驱动程序】路径，单击右键，在弹出的快捷菜单中选中【新建驱动程序选项】，弹出【新建 JDBC 驱动程序】如图 4-12 所示。



图 4-12 添加数据库驱动

单击【添加】按钮，在弹出的【文件】浏览对话框中选中要添加的驱动程序的 Jar 文件，则【驱动程序类】和【名称】输入框将被自动填写。单击【确定】按钮，MySQL 数据库的驱动程序添加到运行环境中。

（2）创建数据库连接类 `DBConnection.java`。

（3）在 Java 代码中定义数据源。具体操作如下：在 Java 代码 `DBConnection` 的编辑视图下，单击

右键，在弹出的快捷菜单中选择【企业资源】→【使用数据库】，弹出【选择数据库】对话框，如图4-13所示。



图 4-13 创建数据源

在下拉菜单中选择已经建立的数据源或者选择【新建数据源】创建一个新的数据源选项。注意要选择【创建服务器资源】单选框。最后单击【确定】，则数据源创建完毕，并且 Java 代码中自动生成了获取数据源的代码。查看与 web.xml 处在同一位置下的 context.xml 文件，可以看到已经为数据源自动配置了连接池。

(4) 在 Java 代码中利用数据源获取连接。

(5) 创建 JSP 页面 test.jsp 利用数据源获取的连接访问数据库信息。

示例代码

程序 4-8: DBConnection.java

```
package sample;

import java.sql.Connection;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;

public class DBConnection {

    /** Creates a new instance of DBConnection */
    public DBConnection() {
    }

    public Connection getConnection(){
        try{
            return getMYSQL().getConnection();
        }catch(Exception e){
            System.out.println(e.toString());
        }
        return null;
    }
}
```



```
private DataSource getMYSQL() throws NamingException {
    Context c = new InitialContext();
    return (DataSource) c.lookup("java:comp/env/MYSQL");
}
```

程序 4-9: test.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ page import="sample.DBConnection" %>
<%@ page import="java.sql.*" %>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>JSP Page</title>
</head>
<body>
    <%
        java.sql.Statement sqlStmt=null; //语句对象
        java.sql.ResultSet sqlRst=null; //结果集对象

        DBConnection db=new DBConnection();
        java.sql.Connection c=db.getConnection();
        if(c!=null){
            sqlStmt=c.createStatement(java.sql.ResultSet.TYPE_SCROLL_INSENSITIVE,
                java.sql.ResultSet.CONCUR_READ_ONLY);
            //执行 sql 语句
            String sqlQuery="select id,name,address from users";
            sqlRst=sqlStmt.executeQuery(sqlQuery);

        }
        if(sqlRst!=null){
            %>
            <center>用户信息表</center>
            <table border="1" width="70%" bordercolorlight="#CC99FF" cellpadding="2"
                bordercolordark="#FFFFFF" cellspacing="0">
                <tr>
                    <td align="center">&nbsp;ID</td>
                    <td align="center">姓名</td>
                    <td align="center">地址电话</td>
                </tr>
                <% while (sqlRst.next()) { //取得下一条记录%>
                <tr><!--显示记录-->
                    <td align="center"><%=sqlRst.getString("id") %></td>
                    <td align="center"><%=sqlRst.getString("name") %></td>
                    <td align="center"><%=sqlRst.getString("address") %></td>
                </tr>
                <% } %>
            </table>
```

```

<%
//释放结果集对象
sqlRst.close();
//释放语句对象
sqlStmt.close();
//释放数据库连接
c.close();
}%>
</body>
</html>

```

程序 4-10: context.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<Context path="/News">
  <Logger className="org.apache.catalina.logger.FileLogger" prefix="News."
suffix=".log" timestamp="true"/>
  <Resource auth="Container" name="MYSQL" type="javax.sql.DataSource"/>

  <ResourceParams name="MYSQL">
    <parameter>
      <name>factory</name>
      <value>org.apache.commons.dbcp.BasicDataSourceFactory</value>
    </parameter>
    <parameter>
      <name>driverClassName</name>
      <value>com.mysql.jdbc.Driver</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>jdbc:mysql://localhost:3306/test</value>
    </parameter>
    <parameter>
      <name>username</name>
      <value>root</value>
    </parameter>
    <parameter>
      <name>password</name>
      <value>javaee</value>
    </parameter>
    <parameter>
      <name>maxActive</name>
      <value>20</value>
    </parameter>
    <parameter>
      <name>maxIdle</name>
      <value>10</value>
    </parameter>
    <parameter>
      <name>maxWait</name>

```



```

        <value>-1</value>
    </parameter>
</ResourceParams>
</Context>

```

运行结果

应用部署成功后, 在浏览器的地址栏输入 “http://localhost:9080/News/test.jsp”, 将得到如图 4-14 所示的运行结果。



图 4-14 利用连接池和数据源访问数据库信息

讨论与思考

无论是建设企业内部信息系统还是开发基于 Web 的应用软件, 都离不开数据库的支持。

在 Java EE 架构下, 实现数据库访问的方法主要有以下几种:

- (1) 基于 JDBC 直接访问数据库;
- (2) 基于连接池访问数据库;
- (3) 基于实体 Bean 访问数据库;
- (4) 基于 ORM 工具访问数据库。

本例程主要演示前两种方法, 基于实体 Bean 访问数据库的方法将在例程 6-1 中演示, 基于 ORM (Object Relational Mapping, 对象关系映射) 工具访问数据库的方法将在例程 6-3 中演示。

JDBC 是 Sun 公司制定的基于 Java 数据接口的规范, 作为 Java EE 规范的一部分, 它主要提供 3 项功能: 连接数据库、向数据库发送 SQL 语句和处理返回的结果。这种技术是在需要访问数据库时通过 JDBC 驱动程序建立与数据库的物理连接, 访问结束又断开连接。

数据库连接池的基本思想就是为数据库连接建立一个“缓冲池”。预先在缓冲池中放入一定数量的连接, 当需要建立数据库连接时, 只需从“缓冲池”中取出一个, 使用完毕之后再放回去, 这就大大提高了系统性能。其实数据库连接池的底层实现还是依赖 JDBC 规范的, 只是增加了对于数据库连接资源的管理功能。

基于实体 Bean 访问数据库的核心思想是将数据库中存储的信息映射到实体 Bean。实体 Bean 与数据库信息之间的映射过程称为持久化。实体 Bean 的持久性可以由 Bean 自己来实现, 称为 BMP (Bean-Managed Persistence, 基于 Bean 的管理持久性), 也可以由 EJB 容器来实现, 称为 CMP (Container-Managed Persistence, 基于容器的管理持久性)。

基于 ORM 工具的数据库访问技术是利用对象关系映射中间件实现程序对象到关系数据库数据的映射。这些对象关系映射中间件通常直接对较底层的 JDBC API 进行封装, 从而可以为高层的使用提供面

向对象的编程接口。目前较流行的 ORM 产品有 Hibernate、TopLink 等。

从系统开发复杂度来考虑,前两种方法要编写 JDBC 代码来实现具体的数据库操作,而第 3 种方法中,JDBC 代码的编写很简单甚至不用编写(如在 CMP Bean 中),开发效率高;从应用环境考虑,实体 Bean 可以连接到不同的数据库,可以在具有不同数据库名的不同环境中部署,并可重复使用,也能集成到分布式环境的应用系统中去。第 4 种方法将数据库结构封装,使程序员可以像使用普通对象一样调用数据库的相关接口,从而实现数据库的相关操作,更适合于灵活的数据库移植与更新。因此,前两种方法更适合于小型的 Web 应用;第 4 种方法适合大型的 Web 应用系统,而第 3 种方法更适合分布式环境中多个用户同时更新数据库的情况。

知识点索引

数据库; JDBC; JDBC-ODBC; 数据源; 连接池。

例程 4-3: 创建基于 XML 的网上论坛

目的

- (1) 演示在 Java EE 中如何操作 XML 数据
- (2) JSP 与 JavaBean 的结合

问题

如何建立一个小型的网上论坛,要求实现以下功能:

- (1) 显示发帖列表;
- (2) 显示指定帖子的回复列表;
- (3) 发布新的帖子和回复;
- (4) 显示发帖人的个人信息;

并且要求论坛尽可能的简易灵活,便于部署。

解决方案

为使论坛尽可能的简易,将不采用数据库作为论坛信息的持久化存储方式,而是采用 XML 数据文件的形式。系统采用 JSP+JavaBean 的架构形式,利用 JavaBean 来操作存储在 XML 中的论坛信息。

知识链接

DOM 是 Document Object Model 的缩写,是对 XML 文档的内容进行表示的模型。它把 XML 文档看作是一系列节点和节点间的关系,并且把每一个节点都当作一个对象,所以称为文档对象模型。

DOM 是以层次结构组织的节点或信息片断的集合。这个层次结构允许开发人员在树中导航以寻找特定信息。分析该结构通常需要加载整个文档和构造层次结构,然后才能做任何工作。由于它是基于信息层次的,因而 DOM 被认为是基于树或基于对象的。

DOM 是 W3C 推荐的 XML 文件的标准模型,是与编程语言无关的,因此有多种实现。

自版本 1.4 开始,DOM 已经成为 JDK 的一部分。

实现步骤

- (1) 创建代表用户发帖信息的 JavaBean LeaveWord;
- (2) 创建代表所有帖子集合的类 LeaveWords;
- (3) 创建辅助工具类 DateTimeUtility 用来帮助显示时间信息;
- (4) 创建类操作 XML 文件 XMLOperator;
- (5) 创建帖子列表 topiclist.jsp;
- (6) 创建回复列表 replylist.jsp;
- (7) 创建其他相关页面用来实现增加帖子、回复等功能。

示例代码

本例程的所有代码都在工程 XMLFroum 下。

程序 4-11: LeaveWord.java

```
package com.hyl.bbs;  
public class LeaveWord  
{  
  
    private String id;  
    private String username;  
    private String fromwhere;  
    private String posttime;  
    private String homepage;  
    private String email;  
    private String text;  
    private String topic;  
    private String face;  
  
    public LeaveWord()  
    {  
    }  
  
    public void clear()  
    {  
        id = "";  
        username = "";  
        fromwhere = "";  
        posttime = "";  
        homepage = "";  
        email = "";  
        text = "";  
        topic = "";  
        face = "";  
    }  
    public String getEmail()  
    {  
        return email;  
    }  
}
```

```
public void setEmail(String email)
{
    this.email = email;
}
public String getFace()
{
    return face;
}
public void setFace(String face)
{
    this.face = face;
}
public String getFromwhere()
{
    return fromwhere != null ? fromwhere : "";
}
public void setFromwhere(String fromwhere)
{
    this.fromwhere = fromwhere;
}
public String getHomepage()
{
    return homepage != null ? homepage : "";
}
public void setHomepage(String homepage)
{
    this.homepage = homepage;
}
public String getId()
{
    return id;
}
public void setId(String id)
{
    this.id = id;
}
public String getPosttime()
{
    return posttime != null ? posttime : "";
}
public void setPosttime(String posttime)
{
    this.posttime = posttime;
}
public String getText()
{
    return text != null ? text : "";
}
public void setText(String text)
{

```



```

        this.text = text;
    }
    public String getUsername()
    {
        return username != null ? username : "";
    }
    public void setUsername(String username)
    {
        this.username = username;
    }
    public String toString()
    {
        return "LeaveWord: id='" + id + "' username='" + fromwhere + "' fromwhere='"
            + username + "' posttime='" + posttime + "' homepage='" + homepage + "' email='"
            + email + "' text='" + text + "' topic='" + topic + "' face='" + face + "\"";
    }
    public LeaveWord getOwnObject()
    {
        return this;
    }
    public String getTopic() {
        return topic != null ? topic : "";
    }
    public void setTopic(String topic) {
        this.topic = topic;
    }
}

```

程序说明：作为一个 JavaBean，用来代表对帖子的抽象。

程序 4-12: LeaveWords.java

```

package com.hyl.bbs;

import java.util.Vector;
public class LeaveWords
{
    private Vector leaveWords;
    public LeaveWords()
    {
        leaveWords = new Vector();
    }
    public void addLeaveWord(LeaveWord leaveWord)
    {
        leaveWords.add(leaveWord);
    }
    public int size()
    {
        return leaveWords.size();
    }
}

```

```

public LeaveWord getLeaveWordAt(int index)
{
    int size = leaveWords.size();
    if(index < 0 || index >= size)
        return null;
    else
        return (LeaveWord)leaveWords.elementAt(index);
}
public String toString()
{
    String newline = System.getProperty("line.separator");
    StringBuffer buf = new StringBuffer();
    for(int i = 0; i < leaveWords.size(); i++)
    {
        LeaveWord lw = (LeaveWord)leaveWords.elementAt(i);
        buf.append(lw.toString()).append(newline);
    }
    return buf.toString();
}
public Vector getLeaveWordsById(String id)
{
    Vector vc = new Vector();
    int size = size();
    for(int i = 0; i < size; i++)
    {
        LeaveWord leaveWord = (LeaveWord)leaveWords.elementAt(i);
        if(id.equalsIgnoreCase(leaveWord.getId()))
            vc.add(leaveWord);
    }
    return vc;
}
}

```

程序说明：通过 Vector 对象来代表论坛所有帖子信息的集合。

程序 4-13: DateTimeUtility.java

```

package com.hyl.bbs;
import java.sql.Timestamp;
import java.text.SimpleDateFormat;
import java.util.Date;
public class DateTimeUtility
{
    public DateTimeUtility()
    {
    }
    public static String getFormattedTime(Timestamp date)
    {
        if(date == null)
            return null;
        else

```



```
        return getFormattedDateTime(date).substring(11);
    }
    public static String getFormattedDateTime(Timestamp date)
    {
        if(date == null)
        {
            return null;
        } else
        {
            SimpleDateFormat lFormatTimestamp = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
            return lFormatTimestamp.format(date);
        }
    }
    public static Timestamp getDateTime(String date, String time, Timestamp defaultValue)
    {
        StringBuffer buf = new StringBuffer();
        Timestamp current = defaultValue;
        SimpleDateFormat lFormatDate = new SimpleDateFormat("yyyy-MM-dd");
        SimpleDateFormat lFormatTime = new SimpleDateFormat("hh:mm:ss");
        SimpleDateFormat lFormatTimestamp = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
        if(isEmpty(date) && isEmpty(time))
            return defaultValue;
        if(defaultValue == null)
            current = getCurrentTimeStamp();
        if(isEmpty(date))
            buf.append(lFormatDate.format(current));
        else
            buf.append(date);
        buf.append(" ");
        if(isEmpty(time))
            buf.append(lFormatTime.format(current));
        else
            buf.append(time);
        Timestamp rtn = null;
        try
        {
            rtn = new Timestamp(lFormatTimestamp.parse(buf.toString()).getTime());
        }
        catch(Exception exception) { }
        return rtn;
    }
    public static String getCurTimeStamp()
    {
        SimpleDateFormat lSimpleDateFormat = new SimpleDateFormat();
        lSimpleDateFormat.applyPattern("yyyy-MM-dd HH:mm:ss");
        Timestamp lTime = Timestamp.valueOf(lSimpleDateFormat.format(new Date()));
        return getFormattedDateTime(lTime);
    }
    public static Timestamp getCurrentTimeStamp()
    {

```

```

SimpleDateFormat lSimpleDateFormat = new SimpleDateFormat();
lSimpleDateFormat.applyPattern("yyyy-MM-dd HH:mm:ss");
return Timestamp.valueOf(lSimpleDateFormat.format(new Date()));
}
public static boolean isEmpty(String string)
{
    boolean isEmpty = true;
    if(string != null && !"".equals(string) && string.trim().length() > 0)
        isEmpty = false;
    return isEmpty;
}
}

```

程序说明：作为辅助工具类，实现 Timestamp 对象的显示控制。

程序 4-14: XMLOperator.java

```

package com.hyl.bbs;
import java.io.*;
import java.net.InetAddress;
import java.net.UnknownHostException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import org.w3c.dom.*;
public class XMLOperator
{
    static Document document;
    private boolean validating;
    public XMLOperator()
    {
        validating = false;
    }
    private boolean openXMLDocument(String fileName)
    {
        boolean bool = true;
        try
        {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            factory.setValidating(validating);
            factory.setNamespaceAware(false);
            DocumentBuilder builder = factory.newDocumentBuilder();
            document = builder.parse(new File(fileName));
            document.getDocumentElement().normalize();
        }
        catch (Exception exp)
        {
            exp.printStackTrace();
            bool = false;
        }
    }
}

```



```

    }
    return bool;
}

public LeaveWords readLeaveWordXML(String fileName)
{
    LeaveWords leaveWords = null;
    try
    {
        if(!openXMLDocument(fileName))
            return leaveWords;
        leaveWords = new LeaveWords();
        Node rootNode = document.getFirstChild();
        NodeList nodes = rootNode.getChildNodes();
        int length = nodes.getLength();
        for(int i = 0; i < length; i++)
        {
            Node node = nodes.item(i);
            if(node.getNodeType() == 1 && "leaveword".equalsIgnoreCase(node.getNodeName()))
                leaveWords.addLeaveWord(UnmarshallLeaveWord(node));
        }
    }
    catch(Exception exp)
    {
        exp.printStackTrace();
        return null;
    }
    return leaveWords;
}

public boolean DelLeaveWord(String fileName, String id, String posttime)
{
    boolean bool = false;
    if(fileName == null || fileName.equals("") || id == null || id.equals("") ||
        posttime == null || posttime.equals(""))
        return false;
    try
    {
        if(!openXMLDocument(fileName))
            return false;
        Node rootNode = document.getFirstChild();
        NodeList leaveWordNodes = rootNode.getChildNodes();
        int length = leaveWordNodes.getLength();
        for(int i = 0; i < length; i++)
        {
            Node leaveWordNode = leaveWordNodes.item(i);
            if(leaveWordNode.getNodeType() == 1 && "leaveWord".equalsIgnoreCase(
                leaveWordNode.getNodeName()))
            {
                String id2 = "";
                String posttime2 = "";
                NodeList nodes = leaveWordNode.getChildNodes();

```

```

        int length2 = nodes.getLength();
        for(int j = 0; j < length2; j++)
        {
            Node node = nodes.item(j);
            String nodeName = node.getNodeName();
            if("id".equalsIgnoreCase(nodeName))
                id2 = UnmarshallText(node);
            else
                if("posttime".equalsIgnoreCase(nodeName))
                    posttime2 = UnmarshallText(node);
        }

        if(id2 != null && !id2.equals("") && posttime2 != null && !posttime2.
equals("") && id.equalsIgnoreCase(id2) && posttime.equalsIgnoreCase
Case(posttime2))
        {
            rootNode.removeChild(leaveWordNode);
            saveXML(fileName);
            return true;
        }
    }
}

catch(Exception ex)
{
    ex.printStackTrace();
    bool = false;
}

return bool;
}

public LeaveWord UnmarshallLeaveWord(Node leaveWordNode)
{
    LeaveWord leaveWord = new LeaveWord();
    NodeList nodes = leaveWordNode.getChildNodes();
    int length = nodes.getLength();
    for(int i = 0; i < length; i++)
    {
        Node node = nodes.item(i);
        String nodeName = node.getNodeName();
        String str = UnmarshallText(node);
        if("id".equalsIgnoreCase(nodeName))
            leaveWord.setId(str);
        else
            if("username".equalsIgnoreCase(nodeName))
                leaveWord.setUsername(str);
            else
                if("fromwhere".equalsIgnoreCase(nodeName))
                    leaveWord.setFromwhere(str);
                else

```



```
        if("posttime".equalsIgnoreCase(nodeName))
            leaveWord.setPosttime(str);
        else
            if("homepage".equalsIgnoreCase(nodeName))
                leaveWord.setHomepage(str);
            else
                if("email".equalsIgnoreCase(nodeName))
                    leaveWord.setEmail(str);
                else
                    if("text".equalsIgnoreCase(nodeName))
                        leaveWord.setText(str);
                    else
                        if("topic".equalsIgnoreCase(nodeName))
                            leaveWord.setTopic(str);
                        else
                            if("face".equalsIgnoreCase(nodeName))
                                leaveWord.setFace(str);
                    }
            }
        return leaveWord;
    }

    public String UnmarshallText(Node textNode)
    {
        StringBuffer buf = new StringBuffer();
        NodeList nodes = textNode.getChildNodes();
        int length = nodes.getLength();
        for(int i = 0; i < length; i++)
        {
            Node node = nodes.item(i);
            if(node.getNodeType() == 3)
                buf.append(node.getNodeValue());
        }
        return buf.toString();
    }

    private void writeXML(String fileName)
    {
        try
        {
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            DocumentBuilder builder = factory.newDocumentBuilder();
            document = builder.newDocument();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    public boolean saveXML(String fileName)
    {
        boolean bool = true;
        try
```

```

    {
        TransformerFactory tf = TransformerFactory.newInstance();
        Transformer transformer = tf.newTransformer();
        DOMSource source = new DOMSource(document);
        transformer.setOutputProperty("encoding", "GB2312");
        transformer.setOutputProperty("indent", "yes");
        PrintWriter pw = new PrintWriter(new FileOutputStream(fileName));
        StreamResult result = new StreamResult(pw);
        transformer.transform(source, result);
    }
    catch(TransformerException exp)
    {
        exp.printStackTrace();
        bool = false;
    }
    catch(Exception e)
    {
        e.printStackTrace();
        bool = false;
    }
    return bool;
}

public synchronized boolean insertTopicAtLast(LeaveWord leaveWord, String
fileName, boolean bReOpen)
{
    boolean bool = true;
    try
    {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        factory.setValidating(validating);
        factory.setNamespaceAware(false);
        DocumentBuilder builder = factory.newDocumentBuilder();
        document = builder.parse(new File(fileName));
        document.getDocumentElement().normalize();
        Node rootNode = document.getFirstChild();
        String id = getMaxIdString(rootNode);
        Element leaveWordElement = document.createElement("leaveword");
        Element idElement = document.createElement("id");
        org.w3c.dom.Text textSeg = document.createTextNode(id);
        idElement.appendChild(textSeg);
        leaveWordElement.appendChild(idElement);
        Element usernameElement = document.createElement("username");
        textSeg = document.createTextNode(leaveWord.getUsername());
        usernameElement.appendChild(textSeg);
        leaveWordElement.appendChild(usernameElement);
        String fromwhere = getLocalIpAddress();
        Element fromwhereElement = document.createElement("fromwhere");
        textSeg = document.createTextNode(fromwhere);
        fromwhereElement.appendChild(textSeg);
        leaveWordElement.appendChild(fromwhereElement);
    }
    catch(Exception e)
    {
        e.printStackTrace();
        bool = false;
    }
    return bool;
}

```



```

String posttime = DateTimeUtility.getCurTimeStamp();
Element posttimeElement = document.createElement("posttime");
textSeg = document.createTextNode(posttime);
posttimeElement.appendChild(textSeg);
leaveWordElement.appendChild(posttimeElement);
Element homepageElement = document.createElement("homepage");
textSeg = document.createTextNode(leaveWord.getHomepage());
homepageElement.appendChild(textSeg);
leaveWordElement.appendChild(homepageElement);
Element emailElement = document.createElement("email");
textSeg = document.createTextNode(leaveWord.getEmail());
emailElement.appendChild(textSeg);
leaveWordElement.appendChild(emailElement);
Element textElement = document.createElement("text");
textSeg = document.createTextNode(leaveWord.getText());
textElement.appendChild(textSeg);
leaveWordElement.appendChild(textElement);
Element topicElement = document.createElement("topic");
textSeg = document.createTextNode(leaveWord.getTopic());
topicElement.appendChild(textSeg);
leaveWordElement.appendChild(topicElement);
Element faceElement = document.createElement("face");
textSeg = document.createTextNode(leaveWord.getFace());
faceElement.appendChild(textSeg);
leaveWordElement.appendChild(faceElement);
rootNode.appendChild(leaveWordElement);
bool = saveXML(fileName);
}
catch (Exception exp)
{
    exp.printStackTrace();
    bool = false;
}
return bool;
}

public synchronized boolean insertReplyAtLast(LeaveWord leaveWord, String
fileName, boolean bReOpen)
{
    boolean bool = true;
    try
    {
        DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
        factory.setValidating(validating);
        factory.setNamespaceAware(false);
        DocumentBuilder builder = factory.newDocumentBuilder();
        document = builder.parse(new File(fileName));
        document.getDocumentElement().normalize();
        Node rootNode = document.getFirstChild();

        Element leaveWordElement = document.createElement("leaveword");

```

```

Element idElement = document.createElement("id");
org.w3c.dom.Text textSeg = document.createTextNode(leaveWord.getId());
idElement.appendChild(textSeg);
leaveWordElement.appendChild(idElement);
Element usernameElement = document.createElement("username");
textSeg = document.createTextNode(leaveWord.getUsername());
usernameElement.appendChild(textSeg);
leaveWordElement.appendChild(usernameElement);
String fromwhere = getLocalIpAddress();
Element fromwhereElement = document.createElement("fromwhere");
textSeg = document.createTextNode(fromwhere);
fromwhereElement.appendChild(textSeg);
leaveWordElement.appendChild(fromwhereElement);
String posttime = DateTimeUtility.getCurTimeStamp();
Element posttimeElement = document.createElement("posttime");
textSeg = document.createTextNode(posttime);
posttimeElement.appendChild(textSeg);
leaveWordElement.appendChild(posttimeElement);
Element homepageElement = document.createElement("homepage");
textSeg = document.createTextNode(leaveWord.getHomepage());
homepageElement.appendChild(textSeg);
leaveWordElement.appendChild(homepageElement);
Element emailElement = document.createElement("email");
textSeg = document.createTextNode(leaveWord.getEmail());
emailElement.appendChild(textSeg);
leaveWordElement.appendChild(emailElement);
Element textElement = document.createElement("text");
textSeg = document.createTextNode(leaveWord.getText());
textElement.appendChild(textSeg);
leaveWordElement.appendChild(textElement);
Element topicElement = document.createElement("topic");
textSeg = document.createTextNode(leaveWord.getTopic());
topicElement.appendChild(textSeg);
leaveWordElement.appendChild(topicElement);
Element faceElement = document.createElement("face");
textSeg = document.createTextNode(leaveWord.getFace());
faceElement.appendChild(textSeg);
leaveWordElement.appendChild(faceElement);
rootNode.appendChild(leaveWordElement);
bool = saveXML(fileName);
}
catch(Exception exp)
{
    exp.printStackTrace();
    bool = false;
}
return bool;
}
private String getMaxIdString(Node rootNode)
{

```



```

NodeList nodes = rootNode.getChildNodes();
int length = nodes.getLength();
String id = "1";
for(int i = length - 1; i >= 0; i--)
{
    Node node = nodes.item(i);
    if(node.getNodeType() == 1 && "leaveword".equalsIgnoreCase(node.
getNodeName()))
    {
        NodeList list = node.getChildNodes();
        int len = list.getLength();
        for(int j = 0; j < len; j++)
        {
            Node n = list.item(j);
            String nodename = n.getNodeName();
            if(nodename.equalsIgnoreCase("id"))
            {
                String strMaxId = UnmarshallText(n);
                id = (new Integer((new Integer(strMaxId)).intValue() + 1)).toString();
                return id;
            }
        }
    }
}
return id;
}

private String getLocalIpAddress()
{
    String str = new String();
    try
    {
        InetAddress inet = InetAddress.getLocalHost();
        str = inet.getHostAddress();
    }
    catch(UnknownHostException exp)
    {
        exp.printStackTrace();
    }
    return str;
}
}

```

程序说明：实现对 XML 文件的操作，从中提取信息并填充到 LeaveWords 对象或者将 LeaveWords 对象的信息持久化到 XML 文件中。

程序 4-15: topiclist.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ page import="java.util.Vector" %>
<%@ page import="com.hyl.bbs.*" %>

```

```

<jsp:useBean id="bean" class="com.hyl.bbs.XMLOperator" scope="page" />
<html>
<head>
<title>基于XML的论坛</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<link rel="stylesheet" type="text/css" href="CSS/style.css">
</head>
<body text="#000000">
<table width="80%" border="1" align="center" cellpadding="4" cellspacing="0"
bordercolor="#999999" bgcolor="#CCCCCC">
<tr>
<td colspan="2" background="images/bmbg.gif">
<font color="#000000">论坛首页</font>
<font color="#000000"><a href="newtopic.jsp">发新帖</a></font>

</td>
</tr>
<tr>
<td width="60%" height="20" valign="top" bgcolor="#F0F0F0"><div align="left">
<b>主题</b> </div></td>
<td width="10%" height="20" valign="top" bgcolor="#F0F0F0"><div align="left">
<b>作者</b> </div></td>
<td width="20%" height="20" valign="top" bgcolor="#F0F0F0"><div align="left">
<b>发布时间</b> </div></td>
<td width="10%" height="20" valign="top" bgcolor="#F0F0F0"><div align="left">
<b>回复数</b></div></td>
</tr>
<%

String strAbsPath = new java.io.File(new java.io.File(application.getRealPath
(request.getRequestURI()))).getParent().getParent();
String path1 = strAbsPath + "\\data\\topic.xml";
String path2 = strAbsPath + "\\data\\reply.xml";
LeaveWords leaveWords = bean.readLeaveWordXML(path1);
LeaveWords replyWords = bean.readLeaveWordXML(path2);
int size = leaveWords.size();
String strSize = (new Integer(size)).toString();
for(int i=0; i<size; i++)
{
    LeaveWord leaveWord = leaveWords.getLeaveWordAt(i);
    Vector vectorReply = replyWords.getLeaveWordsById(leaveWord.getId());
    int sizeReply = vectorReply.size();

%>

<tr>
<td width="60%" height="25" valign="top" bgcolor="#F0F0F0"><div align="left">
<a href="replyList.jsp?id=<%=leaveWord.getId()%>" > <%=leaveWord.getTopic()
%></a></div></td>
<td width="10%" height="25" valign="top" bgcolor="#F0F0F0"><div align="left">

```



```

        <%=leaveWord.getUsername()%></div></td>
        <td width="20%" height="25" valign="top" bgcolor="#F0F0F0"><div align="left">
            <%=leaveWord.getPosttime()%></div></td>
        <td width="10%" height="25" valign="top" bgcolor="#F0F0F0"><div align="left">
            <%=sizeReply%></div></td>

    </tr>
<%
    }
%>

```

程序说明：显示帖子列表。

程序 4-16: replylist.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ page import="java.util.Vector" %>
<%@ page import="com.hyl.bbs.*" %>
<jsp:useBean id="bean" class="com.hyl.bbs.XMLOperator" scope="page" />
<html>
<head>
<title>基于 XML 的论坛</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<link rel="stylesheet" type="text/css" href="CSS/ style.css">
</head>
<body text="#000000">
<table width="80%" border="1" align="center" cellpadding="4" cellspacing="0"
bordercolor="#999999" bgcolor="#CCCCCC">
    <tr>
        <td colspan="2" background="images/bmbg.gif">
            <font color="#000000"><a href="topicList.jsp">论坛首页</a></font>
            <font color="#000000"><a href="newtopic.jsp">发新帖</a></font>

        </td>
    </tr>
</table>
<%
    String id=request.getParameter("id");
    if(id==null)id="";
    String strAbsPath = new java.io.File(new java.io.File(application.getRealPath
(request.getRequestURI()))).getParent().getParent();
    String path1 = strAbsPath + "\\data\\topic.xml";
    String path2 = strAbsPath + "\\data\\reply.xml";
    LeaveWords leaveWords = bean.readLeaveWordXML(path1);
    LeaveWords replyWords = bean.readLeaveWordXML(path2);
    Vector theTopic =leaveWords.getLeaveWordsById(id);
    if(theTopic.size(>0){

        LeaveWord leaveWord = (LeaveWord)theTopic.firstElement();
        String imageSrc = "images/" + leaveWord.getFace() + ".bmp";

```

```

%>
|  |  |  |  | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| <div align="left">     姓名: <%=leaveWord.getUsername()%><br>     来自: <%=leaveWord.getFromwhere()%><br>     <img src=<%=imageSrc%>> </div></td>  | <a href="<%= =leaveWord.getHomepage()%>" target=_blank title="<%=leaveWord.getUsername() %>的主页">主页</a>     | | <a href="mailto:<%=leaveWord.getEmail()%>" title="给<%= =leaveWord.getUsername()%>写信">信箱</a> | ID 号是: <%=leaveWord.getId()%>|     | 留言时间: <%=leaveWord.getPosttime()%> |     | <a href="newreply.jsp?id=<%=id%>">回复</a> |     <hr>     <%=leaveWord.getText()%> </td> </tr> <%     Vector vectorReply = replyWords.getLeaveWordsById(leaveWord.getId());     int sizeReply = vectorReply.size();     for(int j=0; j<sizeReply; j++)     {         LeaveWord replyLeaveWord = (LeaveWord)vectorReply.elementAt(j);         imageSrc = "images/" + replyLeaveWord.getFace() + ".bmp";     } %> |  |  | | --- | --- | | <div align="left"><font color="#CC6633">姓名: <%=replyLeaveWord.getUsername()%><br>         来自: <%=replyLeaveWord.getFromwhere()%><br>         <img src=<%=imageSrc%>> </font></div></td>  <font color="#CC6633">|         <a href="<%=replyLeaveWord.getHomepage()%>" target=_blank title="<%=         =replyLeaveWord.getUsername()%>的主页">主页</a>         | | <a href="mailto: <%=replyLeaveWord.getEmail()%>" title="给<%=         =replyLeaveWord.getUsername()%>写信">信箱</a>         | | ID 号是: <%=replyLeaveWord.getId()%>| 留言时间: <%=         =replyLeaveWord.getPosttime()%>         | </font>     <hr>     <font color="#CC6633">回复内容: <br>         <%=replyLeaveWord.getText()%>     </font> </td> </tr> <%     } %> | | | |

```

程序说明: 显示回复信息列表。

程序 4-17: newtopic.jsp

```

<%@ page contentType="text/html; charset=gb2312" %>
<% session.invalidate(); %>
<html>
<head>
<title>发新帖</title>
<link href="CSS/style.css" rel="stylesheet" type="text/css">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body bgcolor="#0099CC" text="#000000">
<table width="80%" border="1" align="center" cellpadding="4" cellspacing="0"
bordercolor="#CCCCCC" bgcolor="#FFFFFF">
  <Form action="addTopic.jsp" method="post" name="form1">
    <tr background="images/bmbg.gif">
      <td colspan="2"><font color="#000000">新帖子</font></td>
    </tr>
    <tr>
      <td width="19%" align="right" bgcolor="#EFEFEF">姓 名: </td>
      <td width="81%" bgcolor="#EFEFEF"> <input type="text" name="username">
        ** </td>
    </tr>
    <tr>
      <td width="19%" align="right" bgcolor="#EFEFEF">主 页: </td>
      <td width="81%" bgcolor="#EFEFEF"> <input type="text" name="homepage"
        value="http://">
      </td>
    </tr>
    <tr>
      <td width="19%" align="right" bgcolor="#EFEFEF">Email: </td>
      <td width="81%" bgcolor="#EFEFEF"> <input type="text" name="email"> </td>
    </tr>
    <tr>
      <td width="19%" align="right" bgcolor="#EFEFEF">标题: </td>
      <td width="81%" bgcolor="#EFEFEF"> <input type="text" name="topic"> </td>
    </tr>
    <tr>
      <td align="right" bgcolor="#EFEFEF">头 像: </td>
      <td bgcolor="#EFEFEF" alt="单击图片选择头像"> 
        
        
        
        
        
        <input name="face" type="hidden" value="1"></td>
    </tr>
    <tr>
        <td width="19%" align="right" valign="top" bgcolor="#E0E0E0">内容: </td>
        <td width="81%" bgcolor="#E0E0E0"> <textarea name="text" cols="60"
        rows="10"></textarea>
        </td>
    </tr>
    <tr>
        <td width="19%" align="right" bgcolor="#E0E0E0"> </td>
        <td width="81%" bgcolor="#E0E0E0"> <input type="submit" name="Submit" value="提交">
        <input type="reset" name="Submit2" value="重填"> </td>
    </tr>
</Form>
</table>
</body>
</html>

```

程序说明：用来发布新的帖子。

程序 4-18: addtopic.jsp

```

<%@ page contentType="text/html; charset=gb2312" %>
<% request.setCharacterEncoding("GB2312");%>
<jsp:useBean id="xmlBean" class="com.hyl.bbs.XMLOperator" scope="session" />
<jsp:useBean id="bean" class="com.hyl.bbs.LeaveWord" scope="session" >
<jsp:setProperty name="bean" property="*" />
</jsp:useBean>

<% session.setMaxInactiveInterval(900);%>
<%
    if (bean!=null && bean.getUsername()!=null)
    {
        String username = bean.getUsername();
        if (!username.equals(""))
        {
            String filename = request.getRealPath("/") + "data\\topic.xml";

            boolean bool = xmlBean.insertTopicAtLast(bean, filename, true);
        }
    }
    String nextpage = "topicList.jsp";
%>
<jsp:forward page="<%=nextpage%>" />

```

程序说明：调用 JavaBean 的 insertTopicAtLast(bean, filename, true)方法将帖子信息添加到 XML 文件中。

运行结果

应用部署成功后，在浏览器的地址栏输入 <http://localhost:8080/Notice-war/NoticeList>，将得到如

图 4-15 所示的运行结果。



图 4-15 程序运行结果

可以看到论坛中的帖子列表，单击帖子的主题，可以进入帖子的回复列表，如图 4-16 所示。



图 4-16 帖子回复列表

此时可以单击顶部的链接【发新帖】来发表新的帖子，如图 4-17 所示。或者单击链接【回复】来对帖子发出新的回复，如图 4-18 所示。

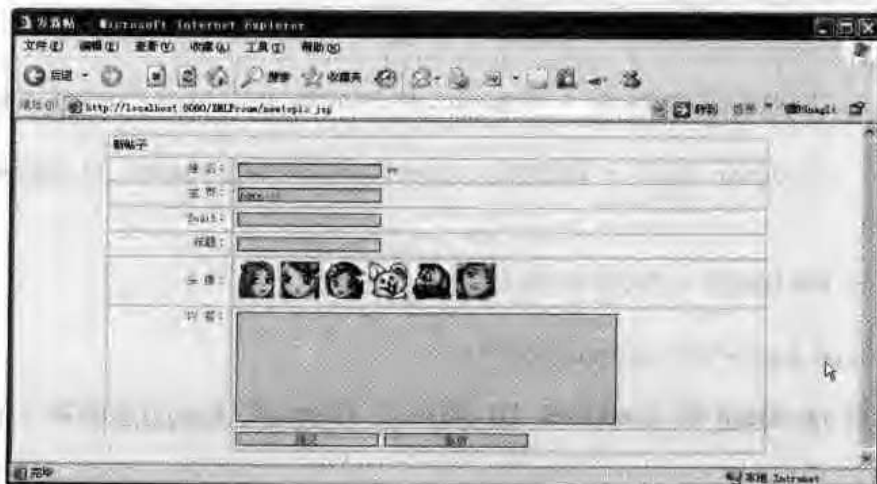


图 4-17 发新帖



图 4-18 新的回复

讨论与思考

本文利用 XML 文件作为持久化存储对象，使用 JAXP 作为编程接口，并利用 DOM 技术对 XML 进行操作。由于 JAXP 已经是 JDK 的组成部分，因此不需要为工程添加专门的 Jar。

首先讨论对 XML 文件的操作技术，目前主要有以下几种。

(1) DOM (文档对象模型)

DOM 为 XML 文档的解析定义了一组接口。解析器读入整个 XML 文档，并构建一个驻留内存的树结构，然后应用代码就可以使用 DOM 接口来操作这个树结构。

优点：整个文档树在内存中，便于操作；支持删除、修改、重新排列等多种功能。

缺点：将整个文档调入内存（包括无用的节点），浪费时间和空间。

使用场合：解析 XML 文档并且还需多次重复访问这些数据；硬件资源充足（内存、CPU）。

(2) SAX

为解决 DOM 的问题，出现了 XML 简单 API (Simple API for XML, SAX)。SAX 属于基于事件驱动的解析。当解析器发现元素开始、元素结束、文本、文档的开始或结束等时，将发送特定的事件消息，程序员可以编写响应这些事件的代码，用来进行适当的处理。

优点：不用事先调入整个文档，占用资源少；SAX 解析器代码比 DOM 解析器代码小，适于内存、处理能力受限制的移动设备及 Applet 等需要远程下载的场合。

缺点：SAX 在内存中没用保留 XML 文档内存的全部映射，它只是根据事件消息进行响应，因此无法对 XML 文档进行重复处理；在消息处理中得到的 XML 文档只是局部的文本，无法知道它在整个文档中的层次结构。

使用场合：Applet；只需 XML 文档的少量内容，很少回头访问；机器内存少。

(3) JAPX

由于开发领域内流行多个 XML 操作接口，为了提高 XML 应用程序的兼容性，又推出了 JAXP 接口规范，它为多个 XML 解析器提供了统一编程接口。这样在应用的底层实现上，即使更换不同的解析器，应用程序页也不用更改代码。因此推荐在应用开发过程中使用 JAPX，可以将代码与各种解析器的实现细节实行隔离。

下面对数据的持久化方案进行讨论。对于数据的持久化，除了基于文件方式外，主要是采用基于关系数据库的方式。基于 XML 的持久化存储是一种新兴的解决方案。基于文件的方式只适合存储简单的

信息，这里主要讨论基于 XML 的持久化存储解决方案与基于关系数据库的解决方案之间的对比。

XML 文件是数据的集合，它是自描述的、可交换的，能够以树状或图形结构描述数据。XML 提供了许多数据库所具备的工具：存储（XML 文档）、模式（DTD, XMLschema, RELAXNG 等）、查询语言（XQuery, XPath, XQL, XML-QL, QUILT 等）、编程接口（SAX, DOM, JDOM）等。但 XML 并不能完全替代数据库技术。XML 缺少作为实用的数据库所应具备的特性：高效的存储、索引和数据修改机制；严格的数据安全控制；完整的事务和数据一致性控制；多用户访问机制；触发器、完善的并发控制等。因此，尽管在数据量小、用户少和性能要求不太高的环境下，可以将 XML 文档用作数据库，但却不适用于用户量大、数据集成度高及性能要求高的作业环境。

随着 Web 技术的不断发展，信息共享和数据交换的范围不断扩大，传统的关系数据库也面临着挑战。数据库技术的应用是建立在数据库管理系统基础上的，各数据库管理系统之间的异构性及其所依赖的操作系统异构性，严重限制了信息共享和数据交换范围；数据库技术的语义描述能力差，大多通过技术文档表示，很难实现数据语义的持久性和传递性，而数据交换和信息共享都是基于语义进行的，在异构应用数据交换时，不利于计算机基于语义自动进行正确数据的检索与应用；数据库属于高端应用，需要昂贵的价格和运行环境。而随着网络和 Internet 的发展，数据交换的能力已成为新的应用系统的一个重要要求。

XML 的好处是数据的可交换性（Portable），同时在数据应用方面还具有如下优点：

- ✎ XML 文件为纯文本文件，不受操作系统、软件平台的限制；
- ✎ XML 具有基于 Schema 自描述语义的功能，容易描述数据的语义，这种描述能为计算机理解和自动处理；
- ✎ XML 不仅可以描述结构化数据，还可有效描述半结构化，甚至非结构化数据。

目前，一种新型的数据库技术——XML 数据库正在兴起，它集成了 XML 技术与关系数据库的优点，感兴趣的读者可以深入探究这一领域。

知识点索引

DOM; JAXP; XML; 持久化存储。

例程 4-4：访问体重检测 Web 服务

目的

演示在 Web 应用中如何调用 Web 服务。

问题

某保健网站需要向其他应用程序提供一个检测体重是否超重的服务。如何实现此功能服务，使得其他应用程序可以很方便地集成？

解决方案

首先基于 JavaBean 创建一标准的 Web 服务，然后创建一 JSP 页面来演示如何调用此 Web 服务。

知识链接

Web 服务技术是应用程序通过内联网或者因特网发布和利用软件服务的一种标准机制。客户程序

(Web 服务用户)可以采用 UDDI (Universal Description, Discovery, and Integration, 统一描述、发现和集成) 协议发现服务器应用程序 (Web 服务供应商) 发布的 Web 服务; 采用 WSDL (Web Services Description Language, Web 服务描述语言) 语言确定服务的接口定义; 采用基于 SOAP 的 XML 文档再通过 HTTP、FTP 和 SMTP 等常用通信方式交换数据。

在 Web 服务的客户应用程序一方, 客户程序在本机调用方法, 但是被调用的方法会被转换为 XML (基于 SOAP) 并通过网络发送给 Web 服务供应商应用程序。供应商再利用 XML 文档 (基于 SOAP) 发回对方法调用的响应。

由于 Web 服务是通过 URL、HTTP 和 XML 得以访问的, 所以运行在任何平台之上、采用任何语言的应用程序都可以访问 Web 服务。

在 Web 服务模型的解决方案中共有三种角色: 服务提供者、服务请求者和服务注册中心, 如图 4-19 所示。其中服务提供者 (服务器) 和服务请求者 (客户端) 是必需的, 服务注册中心是一个可选的角色。它们之间的交互和操作构成了 Web 服务的体系结构。服务提供者定义并实现 Web 服务, 然后将以 WSDL 描述的服务信息发布到服务请求者或服务注册中心; 服务请求者使用 UDDI 查找操作从本地或服务注册中心检索服务描述, 然后使用服务描述与服务提供者进行绑定并调用 Web 服务。这样, 服务请求者就可以调用服务提供者提供的 Web 服务了。

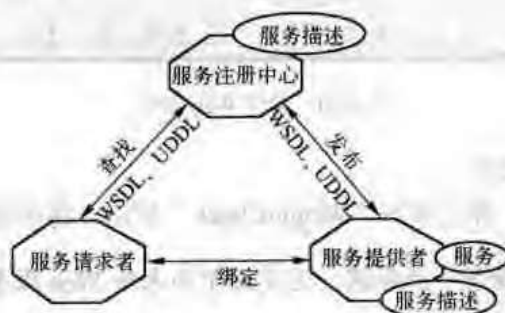


图 4-19 Web 服务模型

Java EE 5 平台提供了 Java API for XML Web Services (JAX-WS) 2.0 来支持 Web 服务开发。作为 Java API for XML-based RPC 1.1 (JAX-RPC) 的后续发行版本, JAX-WS 简化了使用 Java 技术开发 Web 服务的工作。并且通过对多种协议 (如 SOAP 1.1、SOAP 1.2、XML) 的支持, 以及提供用于支持其他协议和 HTTP 的工具, 它解决了 JAX-RPC 1.1 中存在的一些问题。JAX-WS 使用 JAXB 2.0 提供数据绑定服务, 并支持通过定制来控制生成的服务端点接口。通过对标注的支持, JAX-WS 简化了 Web 服务开发, 并缩小了运行时 JAR 文件的大小。

实现步骤

(1) 创建 Web 工程项目 HealthTester。

(2) 在 Web 工程中创建 Web 服务。

① 右键单击 HealthTester 节点, 然后从弹出的快捷菜单中选择【新建】→【Web 服务】。弹出如图 4-20 所示的【新建 Web 服务】对话框。

② 在【Web 服务名称】文本框输入“WeightCheck”, 在【包】文本框中键入“com.hyl”, 然后单击【完成】按钮。将在【项目】目录中新增一个“Web 服务”文件夹, 并显示新建的 Web 服务, 如图 4-21 所示。

IDE 将自动创建服务器所需的部署描述符 (如果存在)。对于 Sun Java System Application Server, 则不需要部署描述符。对于部署到 Tomcat Web Server 的 Web 服务, 将添加 sun-jaxws.xml 及 web.xml

中的 WSServlet 项。



图 4-20 新建 Web 服务

(3) 为 Web 服务添加业务逻辑。

① 展开【Web 服务】节点，然后双击“WeightCheck”节点。将在源代码编辑器中打开 Web 服务。

注意：此时源代码编辑器会报告一个错误。这是由于尚未给 Web 服务添加任何操作的原因。

② 在类主体（位于注释掉的代码之上或之下）中单击鼠标右键，然后在弹出的快捷菜单中选择【Web 服务】→【添加操作】选项。弹出如图 4-22 所示的【添加操作】对话框。



图 4-21 显示新创建的 Web 服务



图 4-22 【添加操作】对话框

③ 在【名称】文本框中输入操作的名称“check”，并在【返回类型】下拉列表中选择“String”作为操作的返回类型。

④ 单击【添加】按钮，创建一个类型为 boolean、名为 sex 的参数。单击【确定】按钮完成添加。

⑤ 按上面的步骤分别添加两个 int 类型的参数 weight 和 height。

⑥ 单击【添加操作】对话框中的【确定】按钮。请注意，check 方法的框架已添加到源代码编辑器中。修改此方法实现业务逻辑：根据输入的性别、体重和身高信息来判断体重是否超标。详细信息如程序 4-19 所示。

(4) 部署 Web 服务：在【项目】目录中右键单击 HealthTester 节点，然后从弹出的快捷菜单中选择【部署项目】。

(5) 测试 Web 服务：在【项目】目录中右键单击 HealthTester 节点，然后从弹出的快捷菜单中选择【运行项目】，打开浏览器，在地址栏中输入“http://localhost:8080/HealthTester/WeightCheckService?Tester”，将得到如图 4-23 所示的运行结果画面。分别在三个输入框中输入 Web 服务 WeightCheck 的操作 check 的三个参数，可以对发布的 Web 服务进行测试。



图 4-23 测试 Web 服务

在浏览器地址栏中输入“http://localhost:8080/HealthTester/WeightCheckService?WSDL”可以查看 Web 服务的 WSDL 文件的内容。可将文件保存到硬盘的某个位置以便供应用程序调用 Web 服务时使用。

(6) 在 Web 应用程序中调用 Web 服务。

① 创建新的 Web 工程 WSCient。

② 创建 Web 服务客户端。

在【项目】目录中选中【WSCient】，单击右键，在弹出的快捷菜单中选择【新建】→【Web 服务客户端】，弹出如图 4-24 所示的【新建 Web 服务客户端】对话框。

创建 Web 服务客户端必须首先选择 Web 服务的接口文件(WSDL 文件)的位置，它可以来自 netbeans 中的一个项目，也可以来自文件系统中的一个文件，或者是网络上的一个 URL 地址。在本例中选择【本地文件】的形式，单击【浏览】按钮选择先前保存的 Web 服务的 WSDL 文件，在【包】文本框输入 Web 客户端文件所在的包“com.hyl.wsclient”，单击【完成】按钮，则 Web 服务客户端创建完毕。在【项目】目录中可以看到新增加的 Web 服务引用，如图 4-25 所示。



图 4-24 【新建 Web 服务客户端】对话框



图 4-25 项目目录中显示的 Web 服务客户端

在项目目录中选中【WSClient】，单击右键，在弹出的快捷菜单中选择【生成项目】，在【文件】目录中可以看到 Netbeans 自动生成的 Web 服务客户端代码，如图 4-26 所示。



图 4-26 文件目录中显示的 Web 服务客户端代码

(7) 在 Web 应用组件中使用 Web 客户端来调用 Web 服务。

① 创建 JSP 页面 invoke.jsp。

② 在源代码编辑器中单击右键，在弹出的快捷菜单中选择【Web 服务客户端资源】→【调用 Web 服务操作】，并在弹出的对话框中选择 Web 操作【check】。netbeans 自动为开发人员生成了调用 Web 服务的框架代码。

③ 创建 JSP 页面 test.jsp，实现用户输入信息的交互。

④ 修改框架代码 invoke.jsp，调用 Web 服务实现业务逻辑。

示例代码

本示例所在的代码分别在工程 HealthTester 和 WSCClient 下。

程序 4-19: WeightCheck.java

```
package com.hyl;

import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;

/**
 *
 * @author hyl
 */
@WebService()
public class WeightCheck {
    /**
     * Web 服务操作
     */
    @WebMethod
    public String check(@WebParam(name = "sex") boolean sex, @WebParam(name = "weight")
        int weight, @WebParam(name = "height") int height) {
        // TODO 实现操作
        int r=height-weight;
        if(sex)r=105-r;
        else r=115-r;
        if(r<-10)return "太瘦";
        if(r<0&&r>-10)return "偏瘦";
        if(r>0&&r<10)return "超重";
        if(r>10&&r<20)return "肥胖";
        else return "严重肥胖";
    }
}
```

程序说明：作为 Web 服务的具体实现者，用来实现 Web 服务的业务逻辑。

程序 4-20: WeightCheckService.wsdl

```
<?xml version="1.0" encoding="UTF-8"?><definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="http://hyl.com/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" targetNamespace="http://hyl.com/"
```



```

name="WeightCheckService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://hyl.com/" schemaLocation="http://
PC364112594285:8080/HealthTester/WeightCheckService/_container$publishing$
subctx/WEB-INF/wsdl/WeightCheckService_schema1.xsd" xmlns:wsdl="http://
schemas.xmlsoap.org/wsdl/" xmlns:soap12="http://schemas.xmlsoap.
org/wsdl/soap12/" />
    </xsd:schema>
  </types>
  <message name="check">
    <part name="parameters" element="tns:check" />
  </message>
  <message name="checkResponse">
    <part name="parameters" element="tns:checkResponse" />
  </message>
  <portType name="WeightCheck">
    <operation name="check">
      <input message="tns:check" />
      <output message="tns:checkResponse" />
    </operation>
  </portType>
  <binding name="WeightCheckPortBinding" type="tns:WeightCheck">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
    <operation name="check">
      <soap:operation soapAction="" />
      <input>
        <soap:body use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>
  <service name="WeightCheckService">
    <port name="WeightCheckPort" binding="tns:WeightCheckPortBinding">
      <soap:address location="http://PC364112594285:8080/HealthTester/
WeightCheckService" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" />
    </port>
  </service>
</definitions>

```

程序说明：Web 服务接口描述文件，其他应用程序根据此接口文件与 Web 服务进行交互。

程序 4-21: test.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
  <head>

```

```

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>体重超标检测</title>
</head>
<body>

<h1>体重超标检测</h1>
<form name="info" action="invoke.jsp" method="POST">
    性别: <select name="sex">
        <option value="true">男</option>
        <option value="false">女</option>
    </select>
    体重: <input type="text" name="weight" value="50" />
    身高: <input type="text" name="height" value="160" />
    <input type="submit" value="测试" name="up" />
</form>

</body>
</html>

```

程序说明: 收集用户信息作为参数来调用 Web 服务。

程序 4-22: invoke.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>测试结果</title>
</head>
<body>
<%
boolean sex = false;
int weight = 48;
int height = 165;
String xb=request.getParameter("sex");
if(xb.equals("true"))sex=true;

weight=Integer.parseInt(request.getParameter("weight"));
height=Integer.parseInt(request.getParameter("height"));
%>
<!-- start web service invocation --%><hr/>
<%
try {
com.hyl.wsclient.WeightCheckService service = new com.hyl.wsclient.WeightCheckService();
com.hyl.wsclient.WeightCheck port = service.getWeightCheckPort();
// TODO initialize WS operation arguments here
// TODO process result here
java.lang.String result = port.check(sex, weight, height);
out.println("您的体重为 "+weight+", 您的身高为 "+height+"<br>");
out.println("测试结果为: "+result);
}

```



```

    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
    %>
    <!-- end web service invocation --%><hr/>
</body>
</html>

```

程序说明：JSP 页面实现了 Web 服务的调用。首先创建一个 Web 服务 WeightCheckService 的示例，然后调用 `getWeightCheckPort()` 方法获取 Web 服务的服务端口，最后根据请求信息中传递来的参数来调用方法 `check()`，并将返回的信息显示出来。

运行结果

说明：由于 Web 服务是由 Web 应用 HealthTester 具体实现的，因此在运行 Web 应用程序之前，确保 Web 应用 HealthTester 处在运行状态。

Web 应用 WSCient 部署完毕并启动后，打开浏览器，在地址栏内输入“`http://localhost:8080/WSCient/test.jsp`”，将得到如图 4-27 所示的运行结果。



图 4-27 参数获取页面

在相应的输入栏内输入信息后，单击【测试】按钮，将得到如图 4-28 所示的运行结果，可以看到 Web 服务已成功调用，并返回处理后的结果信息。



图 4-28 Web 服务调用结果页面

讨论与思考

1. Web 服务的优点

作为一种新兴的应用集成模式，Web 服务具有以下优点。

(1) 跨平台特性

XML 语言本身就是跨平台、跨语言的数据表示方法,再加上通用的 HTTP 协议等,使得 Web 服务天生就适用于基于异构平台的应用。如果用户的客户端包含了各种不同的平台,例如,用户希望其服务既可以被 Java 程序所调用,又可以由 VB 和 COM 程序所调用,则可以有两种选择:一种是为不同的平台提供相应的 API,还要为不同的语言提供 API;如果提供 Web 服务,所有平台和语言都可以调用了!

(2) 穿越防火墙

对于基于二进制协议的 RMI/IIOP 的应用集成,如何穿越不同网络间的防火墙将是一个棘手的问题。客户端浏览器极大可能在 ISP 防火墙后,大多数防火墙都只能允许与外部的 HTTP 连接,因此想要 ISP 防火墙后的客户端能与防火墙外的 RMI/IIOP 的应用端口进行连接的话,就要改变 ISP 的安全策略,让客户端能够连接除了 80 以外的其他端口。可是当运行 RMI/IIOP 的应用的服务器为了安全也在防火墙之后的 DMZ (District of Military Zone, 军事区) 中的话,那这个连接就更加复杂了,要跨越两个防火墙。而 Web 服务由于使用的是 HTTP 协议,传递的是纯文本的 XML 数据,因此拥有穿透防火墙的良好性能。

(3) 对于遗留系统的集成

大多数企业内部都有着各种各样的应用系统,它们是由不同的软件开发商开发的,运行在不同的平台和系统上,系统的开发语言也各不相同。由于现代企业信息自动化要求的提高,各个系统之间的互动和相互通信便提到日程上。因此,保护原有投资,重用遗留系统是当前很多中大型企业的重要任务。

由于遗留系统的运行平台是异构环境,因此企业应用集成的代价一般来说是很高的。但如果使用 Web 服务作为应用集成的手段,将会大大降低集成的消耗。Web 服务与平台和语言无关的特性及其各种平台和环境下的开发工具都是企业应用集成的利器。

2. Web 服务的缺点

但是使用 Web 服务的同时,也要正视它所存在的以下缺点。

(1) Web 服务是无状态的服务

Web 服务还可以说是一种无状态 (Stateless) 的服务。所谓无状态就意味着不保存客户端服务调用者的任何信息。这是由 Web 服务的本质所决定的。Web 服务在本质上是要为应用程序之间提供数据通信的标准,为企业应用之间动态地提供大颗粒度的服务,所以 Web 服务并不适合于非常精细的基于会话的方法调用及复杂的事务 (Transaction) 处理。

(2) 数据绑定不足

Web 服务在数据绑定方面也存在一些不足。因为所有的数据传递都用 XML 格式,因此,需要在二进制数据和 XML 数据之间有个转换。但是,并不是所有的二进制数据都能方便地用 XML 来表示,并不是所有的 JAVA 对象都能被 XML 所表示。因此,经常在转换过程中会出现语义丢失的情况。

(3) 技术要求高,学习曲线较长

Web 服务具有复杂的协议栈和底层实现机制,对于真正意义上严肃的应用,一定要了解 Web 服务的各个方面,设计整体结构和解决方案,还要根据具体的应用调整性能。所有这些都需要对 Web 服务知识的全面掌握。这也就意味着对开发人员的技能和水平提出更高的要求。

(4) 性能上的缺陷

由于 XML 文件的解析处理需要耗费较大的 CPU 计算资源和内存资源,因此对服务器的性能提出更高的要求。网络资源的消耗也是 Web 服务应用的一些限制。因为基于 XML 数据的传递数据量通常要比二进制的协议 (如 RMI/IIOP) 要大得多。这种额外的消耗在网络资源比较紧张或网络传输比较频繁的应用中会产生一定的影响。

3. 提高 Web 服务应用的性能需要注意的事项

要想提高 Web 服务应用的性能,需要对整个系统做全盘考虑。一般来说,有以下几点需要注意。

(1) Web 服务的颗粒度

选择 Web 服务的颗粒度是提高 Web 服务应用的性能的主要手段。因为 Web 服务使用的传输协议为 HTTP 或 SMTP 等, 这些协议都是面向无状态的连接协议, 每一个请求都要建立一个新的连接。因此 Web 服务的调用不能像数据库 JDBC (ODBC) 接口一样可以进行精细而复杂的方法调用 (例如, 先获得 Connection, 再获得结果集, 然后一行一行获取结果)。Web 服务比较适用于大颗粒度的应用, 在一个调用中便获得所有的信息 (比如说银行之间的转账, 在一次调用中就将包括金额和认证等所有的信息都传输过去)。

(2) 谨慎使用 XML 接口

系统之间的接口可以使用 XML, 这样可以增加系统的灵活性; 但不要使用 XML 作为系统内部的接口, 因为这不会带来任何好处, 尽量使用二进制作为系统内部的接口, 避免不必要的 XML 文档的解析和效验; 在处理 XML 的时候, 尽快将 XML 转换成内部对象, XML 的传递只会增加系统的开销。

(3) 最大可能性使用 Cache

当有些信息是只读的, 或者在一段时间内保持不变, 就可以使用 Cache。无论是客户端的 Cache 还是服务器端的 Cache, 都能大大提高系统的性能。

知识点索引

Web 服务; 应用集成。

本章小结

将不同的企业应用集成在一起是一项非常复杂的任务, 因为这些应用由不同的厂商提供, 可能采用不同的编程语言和技术来实现, 运行在不同的操作系统平台上。作为一个企业分布式应用开发标准, Java EE 规范非常重视如何与其他企业信息系统的集成。

应用之间的集成任务如果交由程序人员开发实现将是一项艰巨的任务, 在 Java EE 规范中采用的基本思想是“组件-容器”的编程思想。Java EE 应用的基本软件单元是 Java EE 应用组件。所有的 Java EE 组件都运行在特定的运行环境之中。组件的运行环境被称为容器。Java EE 组件分为 Web 组件和 EJB 组件, 相应地, Java EE 容器也分为 Web 容器和 EJB 容器。

容器为组件提供必需的底层基础功能, 容器提供的底层基础功能被称为服务。组件通过调用容器提供的标准服务来与外界交互。如通过 JNDI 服务来定位到关系数据库的连接, 到 JMS (Java Message Service, Java 消息服务) 消息资料的连接等。

在 Java EE 规范下, Java EE 容器应看作是不同企业信息系统之间的一个中介, 它为 Java 应用访问其他应用系统屏蔽了具体的软件实现细节, 降低了软件开发的难度。Java 组件与容器之间是通过配置文件来实现信息交互的。但是复杂的配置文件同样也给程序开发人员带来一定的困难。在新的 Java EE 5 规范中, 基于资源注入 (Resource Injection) 特性, Java 组件访问企业信息资源不再需要复杂的配置文件, 而是通过 Annotations 标记, 在组件部署到服务器上自动生成。这一特性在本章的多个示例中都有反映。

作为一种应用集成的新技术, Web 服务越来越引起重视, 在新的 Java EE 5 规范中, 提供了 Java API for XML Web Services (JAX-WS) 2.0 来支持 Web 服务开发, 大大降低了开发难度, 提高了开发效率。

第5章 开发 Web 高级功能特性

工欲善其事，必先利其器

——《论语·魏灵公》

在 Web 应用开发中，经常需要实现一些特定的功能特性，如打印、图表、上传、下载等。但上述功能特性往往不是简单地采用某一种 Java EE 技术就能够轻而易举解决的。本章通过多个示例演示如何实现上述功能特性，对于广大 Web 应用开发人员将具有一定的借鉴意义。

例程 5-1：在 Web 页面显示统计图表

目的

WWW 的发展使得基于因特网的应用程序不再局限于静态或者简单的动态内容提供。客户对 Web 应用提出越来越高的要求，如实现桌面应用程序所具有的一些高级 UI 界面和交互特性。统计报表就是 Web 应用程序中经常要使用的一种功能。本节例程重点在于演示并深入探讨在 Web 页面显示统计报表的各种解决方案。

解决方案 1：使用 Applet 显示统计图表

作为客户端的 Java 应用程序，Applet 可以直接调用 Java 的 Swing 图形界面包，利用 Java 本身对图形的支持来显示图表。

问题

在一个在线的水果销售网站中，水果的销售量随季节的变化而变化。管理层对水果销售业绩的统计数据非常关键。如何通过饼形统计图表显示各个季度的水果销量所占的比重？

实现步骤

- (1) 在 Applet 中获取显示的数据信息；
- (2) 在 Applet 中计算要显示的统计信息；
- (3) 在 Applet 的 `paint()` 方法中绘制图表的标题、背景、坐标等辅助信息；
- (4) 在 Applet 的 `paint()` 方法中绘制统计图表；
- (5) 将 Applet 嵌入到网页中进行发布。

示例代码

本示例所在的代码都在工程 `chart` 下。

说明：统计显示的数据，通常是从后台数据库系统动态获取的应用数据。为保持代码的简洁，便于读者理解，本例中的水果销量数据将采用模拟数据的方式。

程序 5-1: cakechart.java

```

//实现图表绘制显示的 Applet 程序
package com.sample;

import java.awt.Color;
import java.awt.Font;
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.RenderingHints;
import java.awt.geom.Arc2D;
import java.awt.geom.Rectangle2D;
import java.util.Date;

public class CakeChart extends java.applet.Applet {
    //初始数据
    private double data[]={311.,650.,713.,427.};
    private String percent[]=new String[data.length]; //每个数据所占百分比
    private int radian[]=new int[data.length]; //每个数据所对应的弧度数
    private double max,min;

    //初始化每块饼的颜色
    private int dia=240; //设置饼图透明度
    private Color c1=new Color(0,255,0,dia);
    private Color c2=new Color(255,255,0,dia);
    private Color c3=new Color(255,0,0,dia);
    private Color c4=new Color(255,128,64,dia);

    private Color colors[]={c1,c2,c3,c4};

    //统计图的宽度和高度
    private int width=600;
    private int height=400;

    //定义标题起始坐标变量
    private int titleStart_x;
    private int titleStart_y;

    //定义圆心坐标
    private int oval_x=60;
    private int oval_y=90;

    //定义椭圆的长轴和短轴
    private int long_axes=280;
    private int short_axes=200;

    //定义图例区域起点坐标
    private int cutlineRect_x=70;
    private int cutlineRect_y=450;

    //定义图例区域矩形的宽度和高度
    private int cutlineRect_width=120;
    private int cutlineRect_height=180;

    //定义日期变量
    private Date date=new Date();

    //定义统计图其他显示信息的超始坐标

```

```

    private int drawDateInfo_x;
    private int drawDateInfo_y;
//定义标题\X\Y轴信息
    private String title="水果销量统计 (饼形统计图) ";
    private String drawDateInfo="绘图日期: ";
//双缓冲设置
    private Image offScreenImage =null;
    private Graphics offScreenBuffer =null;

    public CakeChart() {
    }
    public void init() {
        offScreenImage=this.createImage(width,height);
        offScreenBuffer=offScreenImage.getGraphics();
        NumberBudget();
        CoorBudget();
    }
    public void NumberBudget() {
//求数据中的最大值和最小值
        max=data[0];
        min=data[0];
        for(int mm=0;mm<data.length;mm++) {
            if(data[mm]>max)
                max=data[mm];
            if(data[mm]<min)
                min=data[mm];
        }
//对数据进行求和运算
        float allData_sum=0;
        for(int s=0;s<data.length;s++) {
            allData_sum+=data[s];
        }
//计算每个数据占总数的百分比
        for(int p=0;p<data.length;p++) {
            percent[p]=String.valueOf(Math.round(data[p]/allData_sum*100))+"%";
        }
//计算每个数据所对应的弧度数
        for(int r=0;r<data.length;r++) {
            radian[r]=Math.round((float)data[r]/allData_sum*360);
        }
    }
    public void CoorBudget() {
//预算标题信息的起始坐标
        titleStart_x=22;
        titleStart_y=(width/2)-(title.length()*15/2);
//初始化统计图其他显示信息的起始坐标(位置固定)
        drawDateInfo_x=390;
        drawDateInfo_y=5;
    }
}

```



```

public void update(Graphics g) {
    paint(g);
}

public void paint(Graphics g) {
    PaintBackground(offScreenBuffer);
    PaintChart(offScreenBuffer);
    g.drawImage(offScreenImage, 0, 0, this);
}

public void PaintBackground(Graphics g) {
//渐变背景初始颜色
    Color BackStartColor=Color.white;
    Color BackLastColor=new Color(162,189,230);
    Color titleColor=Color.black;
    Color otherInfoColor=new Color(41,78,118);
//标题背景颜色
    Color titleBackColor=new Color(147,179,225);
//统计图中心区域颜色
    Color cutlineColor=new Color(0,128,255,50);
//图例数据颜色
    Color cutDataColor=Color.white;

    Font titleFont=new Font("黑体",Font.BOLD,18);
    Font otherFont=new Font("宋体",Font.PLAIN,12);
    String year="";
    String month="";
    String day="";
    Graphics2D g2=(Graphics2D)g;
    RenderingHints hints = new RenderingHints(null);
    hints.put(RenderingHints.KEY_ANTIALIASING,RenderingHints.VALUE_ANTIALIAS_ON);
    g2.setRenderingHints(hints);

    GradientPaint gradient=new GradientPaint(0,0,BackStartColor,0,400,
    BackLastColor,false);
    g2.setPaint(gradient);
    Rectangle2D rect=new Rectangle2D.Double(0,0,width,height);
    g2.fill(rect);
//绘制标题背景
    g2.setColor(titleBackColor);
    g2.fill3DRect(0,0,width,30,true);
//绘制图例区域背景
    g2.setColor(cutlineColor);
    g2.fillRect(cutlineRect_y,cutlineRect_x,cutlineRect_width,cutlineRect_height);
    g2.setColor(Color.white);
    g2.drawRect(cutlineRect_y,cutlineRect_x,cutlineRect_width,cutlineRect_height);
//绘制统计图标题
    g2.setFont(titleFont);
    g2.setColor(titleColor);
    g2.drawString(title,titleStart_y,titleStart_x);
}

```

```

//显示统计图其他信息
    g2.setFont(otherFont);
    g2.setColor(otherInfoColor);
    g2.drawString(drawDateInfo,drawDateInfo_y,drawDateInfo_x);
//显示绘制日期
    year=Integer.toString(1900+date.getYear());
    month=Integer.toString(date.getMonth()+1);
    day=Integer.toString(date.getDate());
    g2.drawString(year+"年"+month+"月"+day+"日",drawDateInfo_y+60,drawDateInfo_x);
//显示数据百分比
    int colorRectWH=15;
    int space=5; //图例中小色块之间的间隔距离
    int addData=cutlineRect_x;
    for(int i=0;i<data.length;i++) {
        g2.setColor(colors[i]);
        g2.fill3DRect(cutlineRect_y,addData,colorRectWH,colorRectWH,true);
        if(data[i]==max || data[i]==min)
            g2.setColor(Color.red);
        else
            g2.setColor(cutDataColor);
        int temp=i+1;
        g2.drawString(String.valueOf(data[i])+" ("+"temp+"季度)",cutlineRect_y+
            20,addData+colorRectWH+space);
        addData+=colorRectWH+space;
    }
}

public void PaintChart(Graphics g) {
    int start=0;
    int rVal=192;
    int gVal=192;
    int bVal=192;
    int frameCount=60;
    Graphics2D g2=(Graphics2D)g;
    RenderingHints hints = new RenderingHints(null);
    hints.put(RenderingHints.KEY_ANTIALIASING,RenderingHints.VALUE_ANTIALIAS_ON);
    g2.setRenderingHints(hints);

    for(int t=20;t>=0;t--) {
        rVal=t*255/frameCount;
        gVal=t*255/frameCount;
        bVal=t*255/frameCount;
        g2.setColor(new Color(rVal,gVal,bVal,50));
        g2.drawOval(oval_x,oval_y+t,long_axes,short_axes);
    }

    for(int a=0;a<data.length;a++) {
        Arc2D arc = new Arc2D.Float(Arc2D.PIE);
        g2.setColor(colors[a]);
        arc.setFrame(oval_x,oval_y,long_axes,short_axes);
        arc.setAngleStart(start);
    }
}

```



```

        arc.setAngleExtent(radian[a]);
        g2.fill(arc);
        if(data[a]==max || data[a]==min)
            g2.setColor(Color.white);
        else
            g2.setColor(new Color(223,223,223,150));
        g2.draw(arc);
        start+=radian[a];
    }
}

```

程序说明：使用 Applet 显示统计图形的关键在于重载 Applet 的 `paint()` 方法，在其中调用 Java 图形界面包 `Swing` 来进行操作，最后输出到页面进行显示。在绘制统计图表之前，首先要计算统计数据，在本例中由 `NumberBudget()` 完成，`CoorBudget()` 主要完成对显示信息的坐标位置的计算。图表的绘制主要由 `PaintBackground(Graphics g)` 和 `PaintChart(Graphics g)`，其中 `PaintBackground()` 负责显示图表的背景、标题等辅助信息，`PaintChart()` 用来绘制饼形图。

值得一提的是，为了提高绘图效率，代码采用了双缓存技术，即首先在内存中生成图像，然后把内存中的图像绑定到屏幕对象进行显示。关于双缓存技术的详细信息，可以查阅更多的参考资料。

程序 5-2: index.html

```

//嵌入 Applet 的 HTML 页面
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html lang='zh'>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=GB2312">
    <title></title>
  </head>
  <body>
    <APPLET CODE="com.sample.CakeChart.class" WIDTH="600" HEIGHT="400">
  </body>
</html>

```

程序说明：页面主要用来嵌入 Applet。嵌入 Applet 的方法就是使用标记 `<Applet>`。其中的属性 `code` 就是指 Applet 类。

需要特别强调 Applet 的 class 文件在部署时的位置。集成开发环境如 NetBeans、MyEclipse、JBuilder 等，通常将 Java 源代码编译后放到项目的 `/web-inf/classes/` 目录下。但是，开发人员应该知道，`/web-inf/` 下的所有文件都是不允许客户端直接访问的，这样做的好处在于保护文件的安全。但是 Applet 与常用的 JSP 页面、Servlet 等服务器组件不同的是，它是必须被下载到客户端执行的。因此，如果将应用直接在集成环境中发布到应用服务器上执行，就造成 `index.html` 中的 Applet 无法正确加载。解决的办法是将编译好的 class 文件从开发环境放置的目录（对 NetBeans 来说是工程目录下的 `/build/` 目录）复制到与页面同一个目录下。如果 Applet 类位于一个包内，还必须位于正确的包路径下，如本例中，`cakechart` 位于包 `com.sample` 内，则在 `index.html` 所在的目录下建立包路径 `com/sample`，将 `cakechart` 的类文件放置在此路径下，Applet 就可正确加载。

运行结果

将应用按照上面的要求完成部署后,启动服务器,在地址栏中输入“<http://localhost:8080/chart/index.html>”,将得到如图 5-1 所示的运行结果。

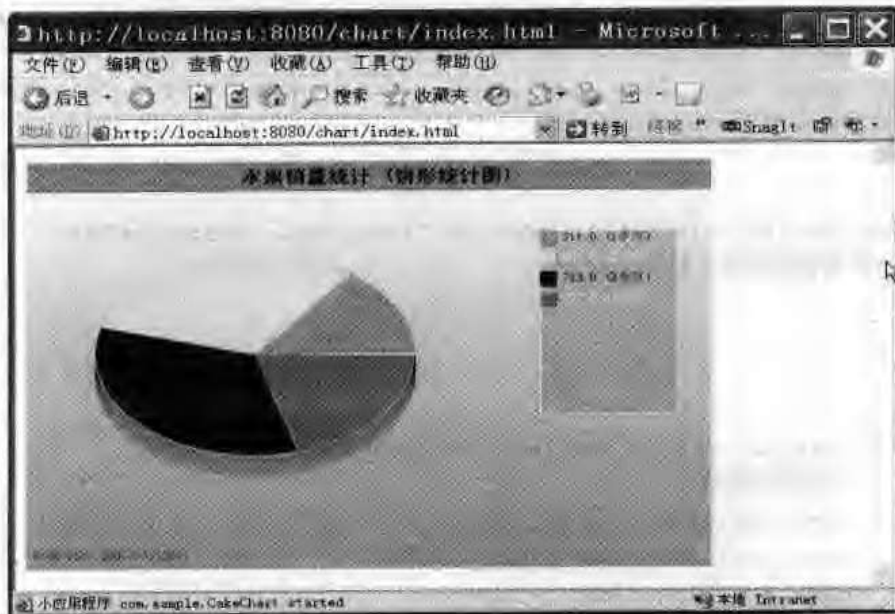


图 5-1 使用 Applet 显示统计图形

解决方案 2: 服务器端的图表解决方案

方案 2 的解决思想是直接在 Web 服务器端生成图表图片文件,然后发送给浏览器。开发人员可以基于一些开源组件来帮助实现这一任务。最常用的开源组件是 JFreeChart。

JFreeChart 是开放源代码站点 SourceForge.net 上的一个 Java 项目。它的功能十分强大,能创建饼图、柱状图(普通柱状图及堆栈柱状图)、线图、区域图、分布图、混合图、甘特图及一些仪表盘,等等,并可生成 PNG 或 JPG 格式的图片文件。

问题

为了分析不同地区水果销量与季节之间的关联关系,现在对水果销量统计图表又提出了新的要求。要求首先显示出各种水果在整体销售量中的比重,又可以详细了解某种水果在各个地区和各季节销量的变化信息。

实现步骤

- (1) 使用 JFreeChart 绘制饼形图来显示水果销量的比重统计信息;
- (2) 使用柱状图显示特定种类水果的销量与地区计季节之间的变化关系。

示例代码

程序 5-3: index.jsp

//显示水果销量的比重分布


```

<%@ page contentType="text/html; charset=GBK"%>
<%@ page import="org.jfree.data.general.DefaultPieDataset"%>
<%@ page import="org.jfree.chart.*"%>
<%@ page import="org.jfree.chart.plot.*"%>
<%@ page import="org.jfree.chart.servlet.ServletUtilities"%>
<%@ page import="org.jfree.chart.labels.StandardPieToolTipGenerator"%>
<%@ page import="org.jfree.chart.urls.StandardPieURLGenerator"%>
<%@ page import="org.jfree.chart.entity.StandardEntityCollection"%>
<%@ page import="java.io.*"%>
<html>
<head>
<meta http-equiv=Content-Type content="text/html; charset=GBK">
<title>水果销售情况统计表</title>
</head>
<body>
<%
DefaultPieDataset data = new DefaultPieDataset();
//数据初始化
data.setValue("火龙果",580);
data.setValue("荔枝",1620);
data.setValue("葡萄",3100);
data.setValue("橙子",8310);
data.setValue("苹果",3520);
data.setValue("哈密瓜",456);

PiePlot3D plot = new PiePlot3D(data); //生成一个 3D 饼图
plot.setURLGenerator(new StandardPieURLGenerator("detailView.jsp"));
//设定图片链接
JFreeChart chart = new JFreeChart("",JFreeChart.DEFAULT_TITLE_FONT, plot, true);
chart.setBackgroundPaint(java.awt.Color.white); //可选, 设置图片背景色
chart.setTitle("水果销售情况统计表"); //可选, 设置图片标题
plot.setToolTipGenerator(new StandardPieToolTipGenerator());
StandardEntityCollection sec = new StandardEntityCollection();
ChartRenderingInfo info = new ChartRenderingInfo(sec);
PrintWriter w = new PrintWriter(out); //输出 MAP 信息

String filename = ServletUtilities.saveChartAsJPEG(chart, 500, 300, info, session);
ChartUtilities.writeImageMap(w, "map0", info, false);

String graphURL = request.getContextPath() + "/servletDisplayChart?
filename=" + filename;

%>
<p align="center">

</p>
</body>
</html>

```

程序说明：在页面的脚本文件中完成了图表的生成及显示准备工作。利用 JfreeChart 生成图表的实现过程主要分为以下几步。

(1) 创建统计图表的数据源 DefaultPieDataset。不同类型的统计图表，具有不同类型的数据源。由于要产生的是饼图，因此采用 DefaultPieDataset，并调用方法 setValue() 为数据源赋值。

(2) 根据数据源创建饼图对象 PiePlot3D，代码如下：

```
PiePlot3D plot = new PiePlot3D(data);
```

(3) 调用 PiePlot3D 的 setURLGenerator() 方法为饼图设置热点链接页面。

(4) 调用 PiePlot3D 的 setToolTipGenerator() 方法为饼图设置提示信息。

(5) 根据 PiePlot3D 生成显示对象 JFreeChart。

(6) 调用 JFreeChart 的 setBackgroundPaint()、setTitle() 等方法进行显示设置。

(7) 创建一个 StandardEntityCollection 集合对象 sec 来包容要绘制的对象。

(8) 利用 sec 创建一个表格绘制对象 ChartRenderingInfo。

(9) 根据 JFreeChart、ChartRenderingInfo 和 session 调用 ServletUtilities 的 saveChartAsJPEG() 方法生成统计图表图像文件。

(10) 调用 ChartUtilities 的 writeImageMap() 方法将图表的热点信息写入表格绘制对象 ChartRenderingInfo。

(11) 最后生成图表图像文件的访问地址 URL。JFreeChart 提供了一个专门用来显示生成图像文件的 Servlet: org.jfree.chart.servlet.DisplayChart，因此，在访问图表图像文件的 URL 中嵌入它的映射地址，并以生成的图像文件的名字作为参数即可。

(12) 将生成的 URL 地址嵌入到页面中。

程序 5-4: detailView.jsp

```
<html>
<head>
  <meta http-equiv=Content-Type content="text/html; charset=GBK">
  <title> 水果销售</title>
</head>

<body>

  <%@ page contentType="text/html; charset=GBK"%>
  <%@ page import="org.jfree.chart.ChartFactory,
  org.jfree.chart.JFreeChart,
  org.jfree.chart.plot.PlotOrientation,
  org.jfree.chart.servlet.ServletUtilities,
  org.jfree.data.category.*"%>
  <%
  CategoryDataset dataset;
  String category=request.getParameter("category");
  category= new String(category.getBytes("ISO8859_1"), "GBK");
  if(category.equals("荔枝")||category.equals("葡萄")||category.equals("苹果")) {
    dataset=getDataSet();
  } else if(category.equals("橙子")||category.equals("哈密瓜")) {
    dataset=getDataSet2();
  }else {
    dataset=getDataSet3();
  }
  %>
```



```

}
String title=category+"网上销售地区与季节情况统计";
JFreeChart chart = ChartFactory.createBarChart3D(title,
    "城市",
    "销量",
    dataset,
    PlotOrientation.VERTICAL,
    true,
    false,
    false);

String filename = ServletUtilities.saveChartAsPNG(chart, 500, 300, null, session);
String graphURL = request.getContextPath() + "/servletDisplayChart?Filename=" + filename;
%>
<p ALIGN="CENTER">
    ">
</p>
<%!
private static CategoryDataset getDataSet() {
DefaultCategoryDataset dataset = new DefaultCategoryDataset();
dataset.addValue(200, "北京", "一季度");
dataset.addValue(180, "上海", "一季度");
dataset.addValue(220, "广州", "一季度");
dataset.addValue(320, "北京", "二季度");
dataset.addValue(350, "上海", "二季度");
dataset.addValue(360, "广州", "二季度");
dataset.addValue(330, "北京", "三季度");
dataset.addValue(340, "上海", "三季度");
dataset.addValue(370, "广州", "三季度");
dataset.addValue(250, "北京", "四季度");
dataset.addValue(280, "上海", "四季度");
dataset.addValue(320, "广州", "四季度");

return dataset;
}
private static CategoryDataset getDataSet2() {
DefaultCategoryDataset dataset = new DefaultCategoryDataset();
dataset.addValue(230, "北京", "一季度");
dataset.addValue(280, "上海", "一季度");
dataset.addValue(120, "广州", "一季度");
dataset.addValue(220, "北京", "二季度");
dataset.addValue(350, "上海", "二季度");
dataset.addValue(260, "广州", "二季度");
dataset.addValue(330, "北京", "三季度");
dataset.addValue(340, "上海", "三季度");
dataset.addValue(370, "广州", "三季度");
dataset.addValue(250, "北京", "四季度");
dataset.addValue(380, "上海", "四季度");
}

```

```

        dataset.addValue(290, "广州", "四季度");
        return dataset;
    }
    private static CategoryDataset getDataSet3() {
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        dataset.addValue(210, "北京", "一季度");
        dataset.addValue(180, "上海", "一季度");
        dataset.addValue(240, "广州", "一季度");
        dataset.addValue(350, "北京", "二季度");
        dataset.addValue(300, "上海", "二季度");
        dataset.addValue(360, "广州", "二季度");
        dataset.addValue(330, "北京", "三季度");
        dataset.addValue(440, "上海", "三季度");
        dataset.addValue(370, "广州", "三季度");
        dataset.addValue(250, "北京", "四季度");
        dataset.addValue(280, "上海", "四季度");
        dataset.addValue(220, "广州", "四季度");
        return dataset;
    }
    %>
</body>
</html>

```

程序说明：页面用来生成柱形图。生成图表的基本思路与上一程序基本相同，只是由于不需要生成热点链接，因此代码也相对简化多了。首先生成数据源，本例中采用的是 `CategoryDataset`，然后调用 `ChartFactory` 的 `createBarChart3D()` 方法来生成图表，随后将图表输出为文件，最后生成图像文件的 URL 地址嵌入到页面中。

运行准备

(1) 下载最新版本的 JFreeChart。下载地址为：<http://sourceforge.net/projects/jfreechart/>。本文使用的版本为 1.0。

(2) 解压文件，将 `jfreechart-1.0.0-rc1/lib` 下的 `jcommon-1.0.0-rc1.jar`，`jfreechart-1.0.0-rc1.jar` 添加到 Web 应用的 lib 目录下。

(3) 在配置文件 `web.xml` 文件中增加以下内容：

```

<servlet>
<servlet-name>DisplayChart</servlet-name>
<servlet-class>org.jfree.chart.servlet.DisplayChart</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>DisplayChart</servlet-name>
<url-pattern>/servletDisplayChart</url-pattern>
</servlet-mapping>

```

说明：JFreeChart 生成的图表都是以图像的形式提供的，为了便于开发人员使用，JFreeChart 提供了一个专门用来显示生成图像文件的 Servlet：`org.jfree.chart.servlet.DisplayChart`。因此，必须将此服务添加到应用的部署文件中，才能够使用它来显示生成的统计图表。

运行结果

将应用按照上面的要求完成部署后,启动服务器,在地址栏中输入“<http://localhost:8080/chart/index.jsp>”,将得到如图 5-2 所示的运行结果。



图 5-2 水果销售比重统计

在图表中单击【橙子】,将得到如图 5-3 所示的销售详细信息图表。

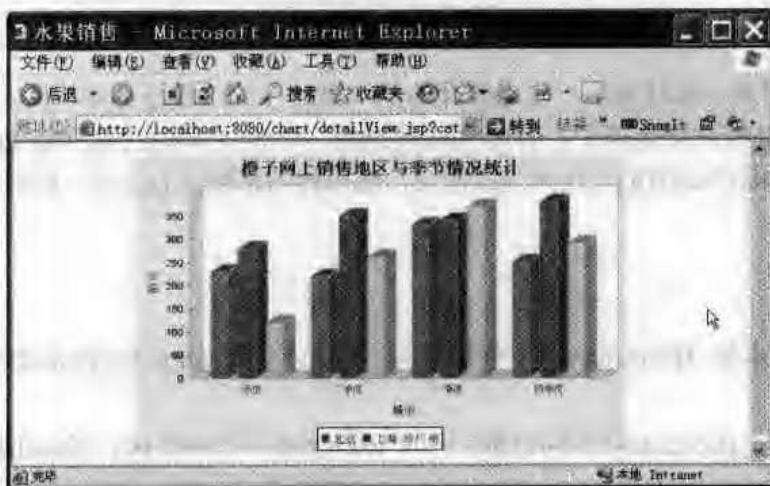


图 5-3 水果销售与地区和季节关联统计

讨论与思考

通过上面例程,基本了解了两种解决方案在 Web 页面显示统计图表的基本步骤和应用效果。使用 Applet 显示图表必须在浏览器端安装 Java 插件,对客户端的要求较高。使用 JfreeChart 在服务器端生成图表图像然后传送到浏览器端显示,对客户端完全没有限制。而且 JfreeChart 还提供了强大的图表显示功能,大大减少了开发人员的工作量。由于 JFreeChart 生成的图表都是以图像的形式提供,则图像可以直接应用与打印,这大大提高了工作效率。

知识点索引

Web 图表; Jfreechart; Applet。

例程 5-2: 为 Web 应用添加打印功能

目的

掌握在 Web 应用中打印报表的方法。

问题

如何打印 Web 应用中的表格?

解决方案 1: 利用 iText 组件打印 Web 表格

iText 是一个开放源码的 Java 类库, 可以用来方便地生成 PDF 文件。利用 iText 组件将 Web 表格转换成 PDF 文件, 利用 PDF 浏览器的打印功能完成 Web 表格的打印。

实现步骤

(1) 在 http://sourceforge.net/project/showfiles.php?group_id=15255&release_id=167948 下载最新版本的类库。本文使用的版本是 1.5.3。注意: 如果生成的 PDF 文件中需要包含中文、日文、韩文字符, 则还需要到 <http://itext.sourceforge.net/downloads/iTextAsian.jar> 下载 iTextAsian.jar 包。

(2) 将下载的 iText 的 Jar 及 iTextAsian.jar 文件添加到工程的类路径下。

(3) 创建 Servlet 组件 PrintServlet, 利用 iText 组件完成打印功能。

示例代码

本例程的所有代码保存在 netbeans 的工程 print 下。

程序 5-5: PrintServlet.java

```
package com.hyl.print;

import com.lowagie.text.Cell;
import com.lowagie.text.Document;
import com.lowagie.text.DocumentException;
import com.lowagie.text.Element;
import com.lowagie.text.Font;
import com.lowagie.text.PageSize;
import com.lowagie.text.Phrase;
import com.lowagie.text.pdf.BaseFont;
import com.lowagie.text.pdf.PdfPTable;
import com.lowagie.text.pdf.PdfWriter;
import java.io.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class PrintServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
```



```

throws ServletException, IOException {
    System.out.println("document.add(BigTable)");
    // step1 创建 Document 对象
    Document document = new Document(PageSize.A4.rotate(), 10, 10, 10, 10); //
    定义纸张类型及方向, 页边距

    try {
        // step 2: 将 Document 与 Servlet 输出绑定
        response.setContentType("application/pdf");
        PdfWriter.getInstance(document, response.getOutputStream());
        // step3
        document.open();
        // step4 定义表格填充内容
        //定义字体
        BaseFont bfChinese = BaseFont.createFont("STSong-Light", "UniGB-
        UCS2-H", BaseFont.NOT_EMBEDDED);
        Font FontChinese = new Font(bfChinese, 12, Font.NORMAL);
        //
        String[] Data = { "0701", "周名", "高工", "2330",
        "2500", "500", "3600" };
        int NumColumns = 7; //定义表格列数

        PdfPTable datatable = new PdfPTable(NumColumns); //创建新表
        int headerwidths[] = { 8, 12, 8, 18, 18, 18, 18 }; // percentage 定义表
        格头宽度
        datatable.setWidths(headerwidths);
        datatable.setWidthPercentage(100); // percentage
        datatable.getDefaultCell().setPadding(3);
        datatable.getDefaultCell().setBorderWidth(2);
        datatable.getDefaultCell().setHorizontalAlignment(Element.ALIGN_CENTER);
        //以下是填充表头
        String[] headerData = { "编号", "姓名", "职称", "基本工资",
        "生活补助", "交通补助", "奖金" };
        for(int k=0;k<7;k++){
            Phrase ph = new Phrase(headerData[k], FontChinese);
            Cell cell = new Cell(ph);

            datatable.addCell(ph);
        }
        datatable.setHeaderRows(1);

        datatable.getDefaultCell().setBorderWidth(1);
        for (int i = 1; i < 300; i++) {
            if (i % 2 == 1) {
                datatable.getDefaultCell().setGrayFill(0.9f);
            }
            for (int x = 0; x < NumColumns; x++) {
                Phrase ph = new Phrase(Data[x], FontChinese);
                // Cell cell = new Cell(ph);
                datatable.addCell(ph);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

```

```

    }
    document.add(datatable); //添加表到 Document
} catch(DocumentException de) {
    de.printStackTrace();
    System.err.println("document: " + de.getMessage());
}

// step 5: 关闭文档
document.close();
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}
}

```

程序说明: Servlet 利用 iText 组件将要打印的文档转换为一个 PDF 文件并输出到客户端, 然后就可以利用客户端 PDF 阅读器的内置打印功能实现文档的打印。具体步骤如下: 首先创建一个 Document 对象, 它代表内存中的一个 PDF 文件; 然后调用 HttpServletResponse 对象的 setContentType("application/pdf")方法设置 Servlet 组件的响应类型为 PDF 文件; 随后调用 PdfWriter 对象的 getInstance()方法将 Document 对象与 Servlet 的输出流绑定到一起, 这样, 创建的 PDF 文档将被输出到客户端浏览器并能够自动调用 PDF 阅读器来显示处理。(说明: 如果客户端浏览器没有安装 PDF 阅读器, 则文件将被提示保存到本地客户端。)

下面就可以对 Document 对象进行操作了。第一步调用 open()方法打开 Document 对象, 然后调用 add()方法添加各种对象到文档中。目前版本的 iText 支持添加表格、图像、文本等。在本例中演示的为添加表格对象 PdfPTable。最后调用 close()方法关闭 Document 对象。

需要特别指出的是, 如果需要在 PDF 中输出中文信息, 则必须创建支持中文的字体, 然后利用这种字体来创建 PDF 显示组件, 如本例中先创建支持中文的 FontChinese, 然后以此为参数创建 Phrase 对象。

运行准备

在程序运行之前, 在客户端机器安装 PDF 阅读器 Adobe Acrobat Reader。Adobe Acrobat Reader 是一个查看、阅读和打印 PDF 文件的最佳工具, 而且它是免费的。它的下载地址为 <http://www.adobe.com/cn/products/acrobat/>。

运行结果

程序发布成功后, 在浏览器地址栏输入 “http://localhost:8084/Print/PrintServlet”, 将得到如图 5-4 所

示的运行结果。

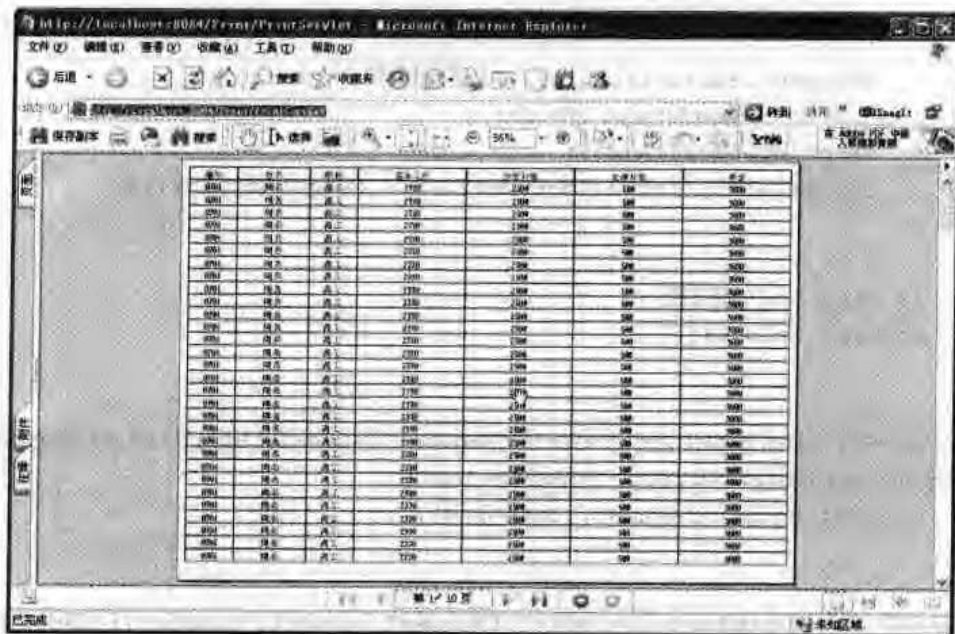


图 5-4 利用 iText 组件实现打印功能

解决方案 2: 利用 JavaScript 脚本打印 Web 报表

在 JavaScript 中, 可以调用浏览器对象 IEWebBrowser, 其 execWB() 方法可实现 Web 页面的打印功能。如果当前 Web 页面中显示的内容不需要完全打印, 则可利用 CSS 来进行控制, 以获得更好的打印效果。

实现步骤

- (1) 在页面中通过语句 `<OBJECT id=WebBrowser classid= CLSID:8856F961-340A-11D0-A96B-00C04FD705A2 height=0 width=0></OBJECT>` 引入浏览器对象 IEWebBrowser。
- (2) 定义控制打印的样式表, 以便获得更好的打印效果。
- (3) 调用 JavaScript 脚本 `document.all.WebBrowser.ExecWB(7,1)` 实现打印。

示例代码

程序 5-6: print.html

```
<html>
<head>
<meta name=vs_targetSchema content="http://schemas.microsoft.com/intellisense/ie5">
<title>利用 JavaScript 实现打印</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<!--media=print 这个属性可以在打印时有效-->
<style media=print>
.NoPrint{display:none;}
.PageNext{page-break-after:always;}
</style>
```

```

<style>
.tab
{
border-color: #000000 #000000 #000000 #000000;
border-style: solid;
border-top-width: 1px;
border-right-width: 1px;
border-bottom-width: 1px;
border-left-width: 1px;
}
</style>
<OBJECT id=WebBrowser classid=CLSID:8856F961-340A-11D0-A96B-00C04FD705A2 height=0
width=0></OBJECT>
</head>
<body>

<hr align="center" width="90%" size="1" noshade class="NOPRINT">
<table border="1">
  <thead>
    <tr>
      <th>编号</th>
      <th>姓名</th>
      <th>职称</th>
      <th>基本工资</th>
      <th>生活补贴</th>
      <th>交通补助</th>
      <th>奖金</th>
    </tr>
  </thead>
  <tbody>

    <tr>
      <td>0701</td>
      <td>周名</td>
      <td>高工</td>
      <td>2330</td>
      <td>2500</td>
      <td>500</td>
      <td>3600</td>
    </tr>
    <tr>
      <td>0701</td>
      <td>周名</td>
      <td>高工</td>
      <td>2330</td>
      <td>2500</td>
      <td>500</td>
      <td>3600</td>
    </tr>
    <tr>

```


<td>0701</td>
<td>周名</td>
<td>高工</td>
<td>2330</td>
<td>2500</td>
<td>500</td>
<td>3600</td>
</tr>
<tr>
<td>0701</td>
<td>周名</td>
<td>高工</td>
<td>2330</td>
<td>2500</td>
<td>500</td>
<td>3600</td>
</tr>
<tr>
<td>0701</td>
<td>周名</td>
<td>高工</td>
<td>2330</td>
<td>2500</td>
<td>500</td>
<td>3600</td>
</tr>
<tr>
<td>0701</td>
<td>周名</td>
<td>高工</td>
<td>2330</td>
<td>2500</td>
<td>500</td>
<td>3600</td>
</tr>
<tr>
<td>0701</td>
<td>周名</td>
<td>高工</td>
<td>2330</td>
<td>2500</td>

```

        <td>500</td>
        <td>3600</td>
    </tr>
    <tr>
        <td>0701</td>
        <td>周名</td>
        <td>高工</td>
        <td>2330</td>
        <td>2500</td>
        <td>500</td>
        <td>3600</td>
    </tr>
    <tr>
        <td>0701</td>
        <td>周名</td>
        <td>高工</td>
        <td>2330</td>
        <td>2500</td>
        <td>500</td>
        <td>3600</td>
    </tr>
</tbody>
</table>

<!--分页-->
<div class="PageNext"></div>
<hr align="center" width="90%" size="1" noshade class="NOPRINT">
<table border="1">
    <thead>
        <tr>
            <th>编号</th>
            <th>姓名</th>
            <th>职称</th>
            <th>基本工资</th>
            <th>生活补贴</th>
            <th>交通补助</th>
            <th>奖金</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>0702</td>
            <td>胡军</td>
            <td>高工</td>
            <td>2330</td>
            <td>2500</td>
            <td>500</td>
            <td>3600</td>
        </tr>
    </tbody>
</table>

```


	0702	胡军	高工	2330	2500	500	3600
	0702	胡军	高工	2330	2500	500	3600
	0702	胡军	高工	2330	2500	500	3600
	0702	胡军	高工	2330	2500	500	3600
	0702	胡军	高工	2330	2500	500	3600

```

        <td>2330</td>
        <td>2500</td>
        <td>500</td>
        <td>3600</td>
    </tr>
    <tr>
        <td>0702</td>
        <td>胡军</td>
        <td>高工</td>
        <td>2330</td>
        <td>2500</td>
        <td>500</td>
        <td>3600</td>
    </tr>
    <tr>
        <td>0702</td>
        <td>胡军</td>
        <td>高工</td>
        <td>2330</td>
        <td>2500</td>
        <td>500</td>
        <td>3600</td>
    </tr>
    <tr>
        <td>0702</td>
        <td>胡军</td>
        <td>高工</td>
        <td>2330</td>
        <td>2500</td>
        <td>500</td>
        <td>3600</td>
    </tr>
</tbody>
</table>
<div class="PageNext"></div>
<hr align="center" width="90%" size="1" noshade class="NOPRINT">
<input type="button" value="打印预览" onclick=document.all.WebBrowser.ExecWB(7,1)
class="NOPRINT">
</body>
</html>

```

程序说明：首先定义了样式表来控制打印效果，将不需要打印的对象过滤。

```

<style media=print>
.Noprnt{display:none;}
.PageNext{page-break-after:always;}
</style>

```

利用语句<OBJECT id=WebBrowser classid=CLSID:8856F961-340A-11D0-A96B-00C04FD705A2 height=0

width=0></OBJECT>引入浏览器对象IEWebBrowser, 调用JavaScript脚本document.all.WebBrowser.ExecWB(7,1)实现预览和打印。

运行结果

程序发布成功后, 在浏览器地址栏输入“http://localhost:8084/Print/print.html”, 将得到如图 5-5 所示的运行结果。



图 5-5 Web 页面中的表格数据

单击页面底部的【打印预览】按钮, 可以对页面的打印效果进行预览, 如图 5-6 所示。并可单击预览窗口左上角的【打印】按钮可打印页面。



图 5-6 利用 JavaScript 实现打印功能

讨论与思考

打印是 Web 应用开发中经常需要实现的一个功能, 也是一个比较棘手的问题。示例中列举了两种打印解决方案。

利用 iText 组件将要打印的内容输出为 PDF 文件，利用 PDF 阅读器内置的打印功能来实现是一种较常见的解决方案，它可以比较精确地控制打印显示的内容和格式，但缺点也很明显，除了代码编写量较大外，还要求客户端必须安装 PDF 阅读器，这就给 Web 应用的客户端部署带来困难。

利用 JavaScript 脚本调用浏览器内置对象可轻松实现对 Web 页面的打印，借助于 CSS，还可对打印做进一步的控制，但是从打印的效果可以看出，由于控制手段缺乏，打印的效果很难满足要求。在对打印要求不高的情况下可以采用这种方式。

除了上述两种方式外，还有一种较为复杂的打印解决方案，即采用 Applet 来实现。Applet 调用 Swing 组件，可以精确控制页面布局，以及打印分页、套打等高级功能特性，市面上一些常见的商业打印组件大多采用这种方式。但这种解决方案的缺点也很明显，表现在以下两个方面

(1) 安装 Applet 成本巨大

需要下载十几 MB 的文件。Applet 本身可能并不大，但运行 Applet 所需的 JRE 一般至少十几 MB (jre1.4.2, 15.45 MB)，这将大大影响用户体验。

(2) 需要重复向服务器请求数据，效率低

因为 Applet 方案一般采用 HTML 方式呈现数据，但实际打印时，Applet 一般不会用当前 HTML 页的数据来打印，而是向服务器下载数据到 Applet 中来打印。也就是说，打印的话，必须两次请求，一次 HTML 呈现，一次用来打印。

知识点索引

打印: iText; JavaScript; CSS; Servlet。

例程 5-3: 创建国际化的 Web 应用

目的

掌握如何实现 Java EE 服务端程序的国际化解决方案。

问题

如何实现支持多地区语言的企业电子邮件系统，使得它支持不同地区的员工可以很方便地使用？

解决方案 1: 为不同地区创建单独的页面资源

针对不同地区的用户，专门设置单独的一组页面，以当地的语言和表达习惯来显示页面内容。在系统的入口处，设置一个交互页面供用户选择要使用的语言种类，系统将根据用户的选择，自动导入到相应的页面中。

实现步骤

- (1) 开发针对特定地区的动态页面。为保持程序简洁，代码中仅实现用户登录部分页面。
- (2) 开发系统入口页面，供用户选择自己要使用的语言。

示例代码

本例程的所有代码保存在 netbeans 的工程 international 下。

程序 5-7: index.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
<title>选择语言</title>
</head>
<body>
<table>
<tr>
<td colspan=4 bgcolor="black">
<br/>
<center><font face="arial" size=+2 color="white"><b><i>Smart& Safe</i> Email </b>
</font></center>
<br/>
</td>
</tr>
<tr><td>
<c:url value="en/login.jsp" var="englishURL"/>
<a href="${englishURL}">

</a>
</td>
<td>
<c:url value="ja/login.jsp" var="japaneseURL"/>
<a href="${japaneseURL}">

</a>
</td>
<td>
<c:url value="ko/login.jsp" var="koreanURL"/>
<a href="${koreanURL}">

</a>
</td>
<td>
<c:url value="zh/login.jsp" var="chineseURL"/>
<a href="${chineseURL}">

</a>
</td>
</tr>
</table>
</body>
</html>

```

程序说明：程序提供了一个地区和语言选项的用户选择界面，供用户进行选择。

程序 5-8: login.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" >
<title>Smart & Safe 电邮</title>
</head>
<body>
<c:url value="confirm.jsp" var="actionURL"/>
<form action="${actionURL}" method="post">
<table>
<tr>
<td colspan=2 bgcolor="black">
<br/>
<center><font face="arial" size=+2 color="white"><b><i>Smart & Safe</i> 电邮</b>
</font></center>
<br/>
</td>
</tr>
<tr>
<td>用户帐号</td>
<td><input type="text" name="userid" size="40"/></td>
</tr>
<tr>
<td>密码</td>
<td><input type="password" name="pass" size="40"/></td>
</tr>
<tr>
<td colspan="2" align="center">
<input type="submit" value="登录"/></td>
</tr>
</table>
</form>
</body>
</html>

```

程序说明: 位于子目录 zh 下, 作为中文界面的邮件系统登录界面。

程序 5-9: conform.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" >
<title>Smart & Safe 电邮</title>
</head>
<body>

```



```
欢迎 ${param.userid}  
</body>  
</html>
```

程序说明：位于子目录 zh 下，作为中文界面的邮件系统登录确认。

运行结果

应用部署成功后，在浏览器的地址栏输入“http://localhost:8084/international/index.jsp”，将得到如图 5-7 所示的运行结果。



图 5-7 语言选择界面

单击中文图标，将得到如图 5-8 所示的登录界面。单击日文图标，将得到如图 5-9 所示的登录界面。



图 5-8 中文登录界面



图 5-9 日文登录界面

解决方案 2：利用标准标记库自动绑定地区属性资源

创建针对不同地区的属性资源文件，利用 JSTL 在页面中引用资源文件。页面在运行时，将根据用户请求的地区属性，自动绑定相应的属性资源，从而使得页面可以支持不同地区的用户。

运行准备

将 JSTL 类库 (jstl.jar 和 standard.jar) 添加到项目的类路径中。具体操作为：选中项目文件夹，单击右键，在弹出的快捷菜单中选择【添加库】选项，弹出如图 5-10 所示的【添加库】对话框。

在对话框的列表中选择【JSTL 1.1】，单击对话框底部的【添加库】按钮，则成功地将 JSTL 相关的类库添加到项目的类路径中。



图 5-10 【添加库】对话框

实现步骤

(1) 创建属性资源文件 app.properties。Netbeans 为创建属性资源文件提供了很好的工具支持。在【项目】视图中选中【源包】文件夹，单击右键，在弹出的快捷菜单中选择【新建】→【属性文件】，则可创建一个新的属性文件。

(2) 为属性文件添加多地区语言环境支持。为了使应用程序支持不同地区语言环境显示，可以为属性文件添加多地区语言环境支持。具体操作为：选中属性文件，单击右键，在弹出的快捷菜单中选择【添加语言环境】选项，弹出如图 5-11 所示的【新建语言环境】对话框。在【预定义的语言环境】下拉列表中选中应用需要支持的语言环境，然后单击【确定】按钮即可。

(3) 在属性文件中添加属性。选择属性文件，单击右键，在弹出的快捷菜单中选择【添加属性】选项，弹出如图 5-12 所示的【新建属性】对话框。在【键】文本框中输入属性的键值，在【值】文本框中输入属性的值，单击【确定】按钮即可。属性自动转换成 ASCII 码保存在属性文件中。



图 5-11 【添加语言环境】对话框



图 5-12 添加属性信息

说明：在保存后的属性文件中，所有的 Unicode 字符均被 neibans 自动转义为 ASCII 编码。这是因为 Java 的资源绑定机制只接受用 ASCII 编码的属性文件。

示例代码

程序 5-10: index.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head>
<title>语言选择</title>
</head>
<body>
<table>
<tr>
<td colspan=4 bgcolor="black">
<br/>
<center><font face="arial" size=+2 color="white"><b><i><i>Smart& Safe</i> Email
</b></font></center>
<br/>

</td>
</tr>
<tr><td>
<c:url value="login.jsp" var="englishURL">
<c:param name="locale" value="en_US"/>
</c:url>

<a href="{englishURL}">

</a>
</td>
<td>
<c:url value="login.jsp" var="japaneseURL">
<c:param name="locale" value="ja_JP"/>
</c:url>

<a href="{japaneseURL}">

</a>
</td>
<td>
<c:url value="login.jsp" var="koreanURL">
<c:param name="locale" value="ko_KR"/>
</c:url>
```

```

<a href="{koreanURL}">
    
</a>
</td>
<td>
    <c:url value="login.jsp" var="chineseURL">
        <c:param name="locale" value="zh_CN"/>
    </c:url>

    <a href="{chineseURL}">
        
    </a>
</td>
</tr>
</table>
</body>
</html>

```

程序说明：提供操作界面供用户进行语言环境选择。

程序 5-11: login.jsp

```

<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<html>
<c:set var="loc" value="en_US"/>
<c:if test="{!(empty param.locale)}">
    <c:set var="loc" value="{param.locale}"/>
</c:if>
<fmt:setLocale value="{loc}" />

<fmt:bundle basename="app">
<head>
<title>Smart & Safe <fmt:message key="email"/></title>
</head>
<body>

<c:url value="confirm.jsp" var="formActionURL" />

<form action="{formActionURL}" method="post">
<table>
<tr>
<td colspan=2 bgcolor="black">
<br/>
<center><font face="arial" size=+2 color="white"><b>
    <i>Smart & Safe</i> <fmt:message key="email"/>
    </b></font></center>
<br/>

```



```

</td>
</tr>
<tr>
<td><fmt:message key="userid"/></td>
<td>

<input type="hidden" name="locale" value="{loc}"/>
<input type="text" name="userid" size="40"/></td>
</tr>
<tr>
<td><fmt:message key="password"/></td>
<td><input type="text" name="pass" size="40"/></td>
</tr>
<tr>
<td colspan="2" align="center">
<input type="submit" value="<fmt:message key='login'/>"/></td>
</tr>
</table>
</form>
</body>
</fmt:bundle>
</html>

```

程序说明：显示特定地区信息的登录界面。

程序 5-12: confirm.jsp

```

<%@ page pageEncoding="UTF-8" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<fmt:requestEncoding value="UTF-8" />
<html>
<fmt:setLocale value="{param.locale}" />
<head>
<fmt:bundle basename="app">
<title>Smart & Safe <fmt:message key="email"/></title>
</head>
<body>
<%=java.util.ResourceBundle.getBundle("app").getString("welcome")%><br>
<fmt:message key="welcome"/>{param.userid}
</body>
</fmt:bundle>

</html>

```

程序说明：显示用户登录确认信息提示。

运行结果

应用部署成功后，在浏览器的地址栏输入“[http://localhost:8084/international/resbundle /index.jsp](http://localhost:8084/international/resbundle/index.jsp)”，将得到如图 5-13 所示的运行结果。



图 5-13 多地区语言选择界面

若单击【中文】图标，则得到中文语言环境的登录界面，如图 5-14 所示；若单击【English】图标，则得到英文语言环境的登录界面，如图 5-15 所示。



图 5-14 中文登录界面



图 5-15 英文登录界面

讨论与思考

世界经济日益全球化推动了人们对基于 Web 的软件的需求，因为许多国家的用户都能访问 Web 软件。这些用户使用的语言、显示、数据录入、表示和文化需求等都可能存在很大的差异。因此，对于分布式 Java EE 应用程序的国际化支持成为应用开发中一个不可忽视的问题。

J2SE (Java 2 standard Edition) 用“地区”(Locale)的概念进行国际化。在一台机器上，地区代表

用户选择的显示语言（例如，英语或西班牙语），以及日期、时间、货币等方面的格式化约定。通常，由底层操作系统管理地区选项设置，并在运行的时候通过 Java 虚拟机把它传递给 J2SE。

但是对于分布式的 Java EE 应用程序，情况却大不一样了。访问应用的客户位于不同的机器上，可能分布在世界各地，如何使用统一的服务器应用来为不同语言环境的客户提供服务成为一个棘手的问题。

这里给出了两种解决方案：一是针对不同的语言环境提供一组单独的 JSP 页面显示，另一个是分离所有使用的与地区相关的字符串，代之以从资源绑定获得一个特定于地区的字符串，这种方式使用了 J2SE 特定于地区的资源绑定处理方式。

对于第一种方案，它适用与以下情况：主要用一种语言访问，偶尔从其他地区访问；底层表示层的 JSP 变化不是很频繁。但此解决方案的最大不足在于：当需要更新特定语言的 JSP 集时，所有冗余编码的 JSP 集都必须同时更新。对于一个中等规模的项目而言，这会造成冗长的、容易出错的更新。

对于第二种方案，则要灵活得多，它仅需要一组 JSP 页面，节省了服务器上需要占用的空间，同时也大大节省了应用的维护成本。

下面需要特别提一下如何确定用户的地区属性。虽然可以通过检查从用户浏览器传入的 HTTP 头来自动检测用户的地区属性信息，但这种方式是非常不完善的。用户或操作系统可能没有正确地设置这条信息。再加上一些浏览器处理地区的信息存在 Bug，所以就不难理解，为什么本书所有示例让应用程序显式询问用户首选地区是一个更能接受的确定地区值的方法。

通常没有可靠的方法可以预先知道什么样的字体支持引入的客户机。为了节约存储和内存，默认情况下，大多数现代操作系统不会预先安装对所有 Unicode 字符的所有字体的支持。有些浏览器会在第一次访问包含必要字体的 Web 页面时，尝试下载和安装这些字体。用图片来显示外国字符或者国旗是表示语言选择屏幕的一种可行方式。

知识点索引

国际化：JSTL。

例程 5-4：在 Web 应用中实现文件上传

目的

- (1) 文件的流操作；
- (2) 常见上传组件的使用。

问题

在许多 Web 站点应用中都需要为用户提供通过浏览器上传文档资料的功能，例如，上传邮件附件、个人相片、共享资料等。对文件上传功能，在浏览器端提供了较好的支持，只要将 Form 表单的 `enctype` 属性设置为“`multipart/form-data`”即可。那么在 Web 服务器端如何来获取浏览器上传的文件呢？

解决方案 1：利用流操作实现文件上传

解析客户端的请求，获取要上传的文件的名称，然后在客户端浏览器与服务器端建立起文件传输流，将浏览器端的文件上传到 Web 服务器。

实现步骤

(1) 生成上传文件的页面 index.html。通过将页面中 Form 表单的 enctype 属性设置为“multipart/form-data”对服务器发出上传请求。

(2) 生成 Servlet。UploadServlet 实现服务器端文件上传。

示例代码

说明：本例程的所有代码都位于 Netbeans 的项目文件夹 MyUpload 下。

程序 5-13: index.html

//发出上传请求的浏览器页面

```
<html lang='zh'>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=GB2312">
    <title></title>
  </head>
  <body>
    <h1>文件上传 </h1>
    <h2>提示：您的浏览器必须能支持文件上传！</h2>
    <form action="../../UploadServlet" method="POST" enctype="multipart/form-data">
    <div align="left">
      发送文件:<input type="file" size="40" name="upl-file"> </BR>
      <input type="submit" value="开始发送" >
      <input type="reset" value="重 设">
    </div>
    </form>
  </body>
</html>
```

程序说明：程序实现了一个上传文件的页面。它通过一个 Form 表单向服务器发出文件上传请求，需要注意的是，要将 Form 的 enctype 属性设置为“multipart/form-data”，同时在 Form 中添加一个类型为 file 的 input 组件。

程序 5-14: UploadServlet

//实现文件上传的 Servlet 组件
package com.upload;

import java.io.*;
import java.net.*;

import javax.servlet.*;
import javax.servlet.http.*;

```
public class UploadServlet extends HttpServlet {
  public static final int MAX_SIZE = 1024 * 1024*10;
  public static final String FILE_DIR = "D:/Temp/";
  public void doPost(HttpServletRequest req, HttpServletResponse res)
```



```

        throws ServletException, IOException {
    DataInputStream dis = new DataInputStream(req.getInputStream());
    res.setCharacterEncoding("gb2312");
    PrintStream ps = new PrintStream(res.getOutputStream());
    String content = req.getContentType();
    if (content != null && content.indexOf("multipart/form-data") != -1) {
        int reqSize = req.getContentLength();
        byte[] data = new byte[reqSize];
        int bytesRead = 0;
        int totalBytesRead = 0;
        int sizeCheck = 0;
        while (totalBytesRead < reqSize) {
            // check for maximum file size violation
            sizeCheck = totalBytesRead + dis.available();
            if (sizeCheck > MAX_SIZE) {
                ps.println("文件太大不能上传...");
                return;
            }
            bytesRead = dis.read(data, totalBytesRead, reqSize);
            totalBytesRead += bytesRead;
        }
        String dataString = new String(data, "gb2312");
        // 分离 filepath 并赋值
        dataString = dataString
            .substring(dataString.indexOf("filename="));
        String filePath = dataString.substring(0, dataString
            .indexOf("Content-Type:"));
        // 分离 contentType 并赋值
        dataString = dataString.substring(dataString
            .indexOf("Content-Type:") + 1);
        // String contentType = dataString.substring(0, dataString
        //     .indexOf("\n"));
        dataString = dataString.substring(dataString.indexOf("\n") + 1);
        // 分离文件信息 获得最终想要的字节
        dataString = dataString.substring(2,
            dataString.lastIndexOf("\n") - 1);
        dataString = dataString.substring(0,
            dataString.lastIndexOf("\n") - 1);
        writeFile(dataString.getBytes("ISO-8859-1"), FILE_DIR
            + getFileName(filePath));
        ps.println("文件上传成功.....");
    }
}

public boolean writeFile(byte[] data, String path) {
    File f = null;
    FileOutputStream fos = null;
    try {
        f = new File(path);
        f.createNewFile();
    }
}

```

```

        fos = new FileOutputStream(f);
        fos.write(data, 0, data.length);
    } catch (FileNotFoundException e) {
        return false;
    } catch (IOException e) {
        return false;
    } finally {
        try {
            fos.close();
        } catch (IOException e) {
            return false;
        }
    }
    return true;
}

public String getFileName(String arg) {
    String path = arg.substring(arg.indexOf("\\") + 1, arg.lastIndexOf("\\"));
    path = path.substring(path.lastIndexOf("\\") + 1);
    return path;
}
}

```

程序说明：程序主要实现服务器端文件上传功能。其中：`writeFile(byte[] data, String path)` 方法实现将上传字节流写入到服务器的文件系统的功能，`getFileName(String arg)` 方法实现从字节流中解析出上传文件名的功能，上传文件的核心功能由 `doPost()` 方法实现。服务器端上传文件的基本步骤如下。

(1) 调用 `request` 对象的 `getInputStream()` 方法获取客户端请求输入流并以此为参数创建一个 `DataInputStream` 对象。

(2) 调用 `response` 对象的 `getOutputStream()` 方法获取响应对象的输出流并以此为参数创建一个 `PrintStream` 对象。

(3) 调用 `request` 对象的 `getContentType()` 方法获取请求类型字符串，并根据请求类型字符串是否包含 `multipart/form-data` 判断是否为文件上传请求。

(4) 调用 `response` 对象的 `getLength()` 方法获取请求的长度，并以此创建一个字节数组 `data` 来存放请求内容。

(5) 如果上传文件的内容较大，那么 `DataInputStream` 输入流中不可能包含所有的请求内容，则调用 `DataInputStream` 的 `available()` 方法获取当前输入流中数据的大小。

(6) 调用 `DataInputStream` 的 `read` 方法获取当前输入流中数据并存入数组 `data`。

(7) 循环执行步骤 (5) 和 (6) 直到把请求中的数据全部读完。

(8) 根据字节数组 `data` 的内容创建一个 `String` 对象（注意创建时采用正确的编码参数。由于请求页面的编码类型为“gb2312”，则这时也必须采用“gb2312”作为参数来创建 `String` 对象）。

由于在读入的字节流中除了上传的内容外，还有一些附加信息，如上传文件的名称、类型等。具体格式信息可以参考 HTTP 通信协议的相关资料。因此，在将字节流保存为文件之前，必需将这些附加信息除掉。

(9) 调用 `String` 对象的 `substring()` 方法对字节流进行解析，调用方法 `getFileName()` 获取上传文件的名称，并剥离出上传文件的内容。

(10) 调用 `writeFile()` 方法将上传文件的字节写入上传文件路径下。

运行准备

在服务器上建立上传文件的存放路径（在本例中为 d:\temp）。

运行结果

应用部署成功后，在浏览器的地址栏输入“http://localhost:8080/MyUpload/basic /index.html”，将得到如图 5-16 所示的运行结果。

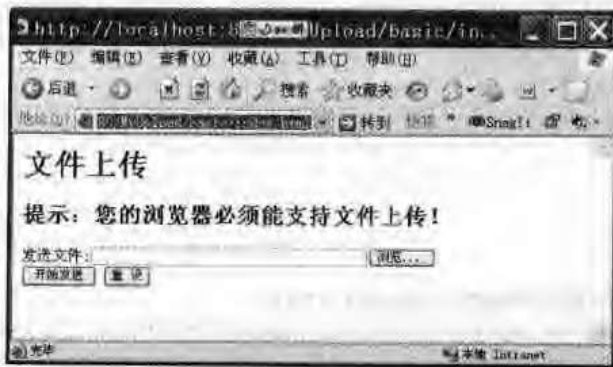


图 5-16 文件上传前端页面

单击【浏览】按钮，选择要上传的文件，然后单击【开始发送】按钮，将得到如图 5-17 所示的运行结果，表明文件已经成功上传。在上传文件路径（本例中为 d:\temp）下，可以看到刚才上传的文件。（注意：本例设置的上传文件的大小限制为 10 MB，超出将不能上传。）

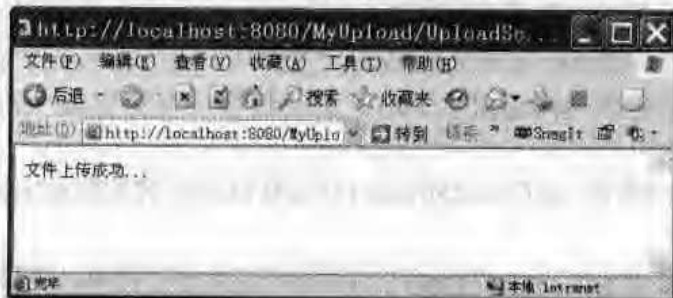


图 5-17 文件上传成功提示信息

解决方案 2：利用 jspSmartUpload 组件实现上传

jspSmartUpload 是由 www.jspsmart.com 网站开发的一个可免费使用的全功能的文件上传下载组件，适于嵌入执行上传下载操作的 JSP 文件中。该组件使用简单，功能强大，利用 jspSmartUpload 组件提供的对象及其操作方法，可以获得全部上传文件的信息（包括文件名、大小、类型、扩展名、文件数据等），存取方便。同时能对上传的文件在大小、类型等方面做出限制。

实现步骤

- (1) 创建上传页面 index.html；
- (2) 创建上传处理页面 upload.jsp，利用 jspSmartUpload 组件实现上传功能。

示例代码

程序 5-15: index.html

```

<html>
<head>
<title>文件上传</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body >
    <p align="center"><font color="#660000" size="5" face="隶书">上传文件选择
    </font></p>
    <form method="post" action="upload.jsp" enctype="multipart/form-data">

        <table width="75%" border="1" align="center">
            <tr><td>
                <div align="center">1、
                <input type="FILE" name="FILE1" size="30"></div>
            </td></tr>
            <tr><td>
                <div align="center">2、
                <input type="FILE" name="FILE2" size="30"></div>
            </td></tr>

            <tr><td>
                <div align="center">
                <input type="submit" name="Submit" value="上传">
                </div>
            </td></tr>
        </table>
    </form>
</body>
</html>

```

程序 5-16: upload.jsp

```

<%@ page contentType="text/html; charset=gb2312" language="java"
import="java.util.*,com.jspsmart.upload.*" errorPage="" %>
<html>
<head>
    <title>文件上传处理页面</title>
</head>
<body >
    <%
        // 新建一个 SmartUpload 对象
        SmartUpload su = new SmartUpload();
        // 上传初始化
        su.initialize(pageContext);
        // 上传文件
        su.upload();
        // 将上传文件全部保存到指定目录
        int count = su.save("/upload/");
        out.println(count+"个文件上传成功! <br>");

        // 逐一提取上传文件信息, 同时可保存文件。
    %>

```



```

for (int i=0;i<su.GetFiles().getCount();i++){
    com.jspSmart.upload.File file = su.GetFiles().getFile(i);
    // 若文件不存在则继续
    if (file.isMissing()) continue;
    // 显示当前文件信息
    out.println("<TABLE BORDER=1>");
    out.println("<TR><TD>表单项名 (FieldName) </TD><TD>"
        + file.getFieldName() + "</TD></TR>");
    out.println("<TR><TD>文件长度 (Size) </TD><TD>"
        + file.getSize() + "</TD></TR>");
    out.println("<TR><TD>文件名 (FileName) </TD><TD>"
        + file.GetFileName() + "</TD></TR>");
    out.println("<TR><TD>文件扩展名 (FileExt) </TD><TD>"
        + file.getFileExt() + "</TD></TR>");
    out.println("<TR><TD>文件全名 (FilePathName) </TD><TD>"
        + file.getFilePathName() + "</TD></TR>");
    out.println("</TABLE><BR>");
}
%>
</body>
</html>

```

程序说明：程序利用 jspSmartUpload 组件实现文件上传功能。首先调用 SmartUpload() 方法创建一个 SmartUpload，然后将隐含对象 pageContext 作为参数调用 initialize() 方法对 SmartUpload 对象进行初始化，随后调用 upload() 方法将文件上传到服务器，最后调用方法 save() 将上传的文件进行保存。Save() 方法的参数为上传文件的存放路径，它是以 Web 应用的根目录作为根目录的。

运行准备

- (1) 在 Web 应用的根目录下创建一个目录 upload 作为上传文件的存放路径；
- (2) 将 jspSmartUpload 组件的 Jar 文件添加到 Web 应用的库文件路径下。

运行结果

应用部署成功后，在浏览器的地址栏输入“http://localhost:8080/MyUpload /smartUpload/index.html”，将得到如图 5-18 所示的运行结果。



图 5-18 选择上传文件

单击【上传】按钮，将得到如图 5-19 所示的运行结果。可以看到文件已经被成功上传。



图 5-19 文件上传成功提示信息

解决方案 3: 利用 common-upload 组件实现上传

common-upload 组件是 Apache 的开源项目之一, 实现了强大的功能, 可以很方便实现将文件上传到服务器上的目录或数据库中。下面将通过代码分别演示如何利用 common-upload 组件上传文件到服务器目录和数据库。

实现步骤

- (1) 生成提交文件上传请求的页面 index.html;
- (2) 生成 Servlet commonUploadServlet 实现服务器端文件上传;
- (3) 生成提交文件上传请求的页面 index2.html;
- (4) 生成 Servlet UpdateLoadDatabaseServlet 实现服务器端文件上传。

示例代码

程序 5-17: index.html

```
<html lang='zh'>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=GB2312">
    <title></title>
  </head>
  <body>
    <h1>文件上传演示</h1>
    <form name="uploadform" method="POST" action="../commonUploadServlet"
    ENCTYPE="multipart/form-data">
      <table border="1" width="450" cellpadding="4" cellspacing="2" bordercolor=
      "#9BD7FF">
        <tr><td width="100%" colspan="2">
          文件 1: <input name="x" size="40" type="file">
        </td></tr>
        <tr><td width="100%" colspan="2">
          文件 2: <input name="y" size="40" type="file">
        </td></tr>
        <tr><td width="100%" colspan="2">
          文件 3: <input name="z" size="40" type="file">
        </td></tr>
      </table>
    </form>
  </body>
</html>
```



```

        </td></tr>
    </table>
    <br/><br/>
    <table>
        <tr><td align="center"><input name="upload" type="submit" value="开始上传"/></td></tr>
    </table>
</form>
</body>
</html>

```

程序说明：程序通过 Form 表单向服务器端发出文件上传请求。

程序 5-18: commonUploadServlet.java

```

package com.upload;

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
import java.util.regex.*;
import org.apache.commons.fileupload.*;

public class commonUploadServlet extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GB2312";
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out=response.getWriter();
        try {
            DiskFileUpload fu = new DiskFileUpload();
            // 设置允许用户上传文件大小,单位为字节,这里设为 2 MB
            fu.setSizeMax(2*1024*1024);
            // 设置最多只允许在内存中存储的数据,单位为字节
            fu.setSizeThreshold(4096);
            // 设置一旦文件大小超过 getSizeThreshold() 的值时数据存放在硬盘的目录
            fu.setRepositoryPath("D:\\temp");
            //开始读取上传信息
            List fileItems = fu.parseRequest(request);
            // 依次处理每个上传的文件
            Iterator iter = fileItems.iterator();

            //正则匹配,过滤路径取文件名
            String regExp=".+\\\\\\\\(.+)$";

            //过滤掉的文件类型
            String[] errorType={".exe",".com",".cgi",".asp"};
            Pattern p = Pattern.compile(regExp);
            while (iter.hasNext()) {
                FileItem item = (FileItem)iter.next();

```

```

//忽略其他不是文件域的所有表单信息
if (!item.isFormField()) {
    String name = item.getName();
    long size = item.getSize();
    if((name==null||name.equals("")) && size==0)
        continue;
    Matcher m = p.matcher(name);
    boolean result = m.find();
    if (result){
        for (int temp=0;temp<errorType.length;temp++){
            if (m.group(1).endsWith(errorType[temp])){
                throw new IOException(name+": wrong type");
            }
        }
        try{

//保存上传的文件到指定的目录
            item.write(new File("d:\\\" + m.group(1)));

            out.print(name+"&nbsp;&nbsp;&nbsp;"+size+"<br>");
        } catch (Exception e){
            out.println(e);
        }

        } else {
            throw new IOException("fail to upload");
        }
    }
} catch (IOException e){
    out.println(e);
} catch (FileUploadException e){
    out.println(e);
}

}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}
}

```


程序说明：程序主要实现了服务器端的文件上传功能。实现上传功能的代码是在方法 processRequest() 中实现。其基本步骤如下：

- (1) 首先创建一个 DiskFileUpload 实例，它是实现文件上传功能的核心组件；
- (2) 调用 DiskFileUpload 的 setSizeMax() 方法设置允许上传文件的最大体积；
- (3) 调用 DiskFileUpload 的 setSizeThreshold() 方法设置允许上传文件存储到服务器上的体积门限，超过此参数的文件将被存放在服务器上的指定位置而不是存放在内存中；
- (4) 调用 DiskFileUpload 的 setRepositoryPath() 方法设置上传文件存储到服务器上的临时路径；
- (5) 调用 DiskFileUpload 的 parseRequest(request) 方法解析请求对象，解析的结果存放在一个列表中；
- (6) 调用 List 的 iterator() 方法将列表中的内存转存到一个枚举对象；
- (7) 从枚举对象中取出值并强制转换为 FileItem 对象，并调用 isFormField() 方法判断是普通的表单项还是上传文件项；
- (8) 如果是上传文件项，则与正则表达式进行匹配，以确认是否是禁止上传的文件类型（本步骤是为了提高应用的安全性，不是必需的）；
- (9) 如果通过匹配，则调用 FileItem 对象的 Write() 方法将文件保存到目录。

程序 5-19: index2.html

```
<html lang='zh'>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=GB2312">
    <title></title>
  </head>
  <body>
    <h1>文件上传演示</h1>
    <form name="uploadform" method="POST" action="../commonUploadServlet"
    ENCTYPE="multipart/form-data">
      <table border="1" width="450" cellpadding="4" cellspacing="2" bordercolor=
      "#9BD7FF">
        <tr><td width="100%" colspan="2">
          文件: <input name="x" size="40" type="file">
        </td></tr>
      </table>
      <br/><br/>
      <table>
        <tr><td align="center"><input name="upload" type="submit" value="开
        始上传"/></td></tr>
      </table>
    </form>
  </body>
</html>
```

程序说明：程序通过 Form 表单向服务器端发出文件上传请求。

程序 5-20: UpdateLoadDatabaseServlet.java

```
package com.upload;

import java.io.*;
```

```
import java.sql.*;
import java.util.Iterator;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
import javax.servlet.*;
import javax.servlet.http.*;
import org.apache.commons.fileupload.DiskFileUpload;
import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.FileUploadException;
public class UpdateLoadDatabaseServlet extends HttpServlet {
    private static final String CONTENT_TYPE = "text/html; charset=GB2312";
    Connection conn=null;
    public void init(ServletConfig config) throws ServletException {
        super.init(config);
        //获取数据库连接
        try{
            Class.forName("org.gjt.mm.mysql.Driver").newInstance();
            conn=
            java.sql.DriverManager.getConnection("jdbc:mysql://localhost/upload","
            root","root");

        } catch (Exception e){
            System.out.println(e.toString());
        }
    }

    protected void processRequest(HttpServletRequest request, HttpServletResponse
    response)
    throws ServletException, IOException {
        response.setContentType(CONTENT_TYPE);
        PrintWriter out=response.getWriter();
        try {
            DiskFileUpload fu = new DiskFileUpload();
            // 设置允许用户上传文件大小, 单位为字节, 这里设为 2 MB
            fu.setSizeMax(2*1024*1024);
            // 设置最多只允许在内存中存储的数据, 单位为字节
            fu.setSizeThreshold(4096);
            // 设置一旦文件大小超过 getSizeThreshold() 的值时数据存放在硬盘的目录下
            fu.setRepositoryPath("D:\\temp");
            //开始读取上传信息
            List fileItems = fu.parseRequest(request);
            // 依次处理每个上传的文件
            Iterator iter = fileItems.iterator();
            //正则匹配, 过滤路径取文件名
            String regExp=".+\\\\\\\\(.+)$";

            //过滤掉的文件类型
            String[] errorType={".exe",".com",".cgi",".asp"};
            Pattern p = Pattern.compile(regExp);
```



```

while (iter.hasNext()) {
    FileItem item = (FileItem)iter.next();
    //忽略其他不是文件域的所有表单信息
    if (!item.isFormField()) {
        String name = item.getName();
        long size = item.getSize();
        if ((name==null||name.equals("")) && size==0)
            continue;
        Matcher m = p.matcher(name);
        boolean result = m.find();
        if (result){
            for (int temp=0;temp<errorType.length;temp++){
                if (m.group(1).endsWith(errorType[temp])){
                    throw new IOException(name+": wrong type");
                }
            }
            try{
                int byteread=0;
                PreparedStatement pstmt=conn.prepareStatement("insert into
uploadfile(name,content) values(?,?)");
                InputStream inStream=item.getInputStream(); //读取输入流,
                也就是上传的文件内容
                pstmt.setString(1,m.group(1));
                pstmt.setBinaryStream(2,inStream,(int)size);
                pstmt.executeUpdate();
                inStream.close();
                out.print(name+"&nbsp;&nbsp;&nbsp;"+size+"<br>");
            } catch (Exception e) {
                out.println(e);
            }

            } else {
                throw new IOException("fail to upload");
            }
        }
    } catch (IOException e){
        out.println(e);
    } catch (FileUploadException e){
        out.println(e);
    }
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

```

```

    }

    public String getServletInfo() {
        return "Short description";
    }
}

```

程序说明：程序利用 common-upload 组件实现上传文件到数据库的功能。上传到服务器目录与上传到数据库的操作基本不同，唯一的区别在最后一步，即上传到服务器目录是调用 FileItem 对象的 write() 方法将文件保存到目录，而上传到数据库是调用 FileItem 对象的 getInputStream() 方法获取上传文件的输入流，然后调用 PreparedStatement 的 setBinaryStream() 方法将输入流数据插入到数据库中。

运行准备

(1) 建立文件上传目录（在本例中为 d:\temp）。

(2) 建立数据库表。

打开 MySQL 数据库，首先建立一个上传文件专用的数据库 upload。系统中仅有 1 个表格（上传文件信息表 uploadfile），用来存储上传文件信息。表格的详细信息分别如表 5-1 所示。

表 5-1 上传文件信息表 uploadfile 的结构信息

字段名	字段类型	备注	字段说明
Id	Integer	主键，系统自增字段	上传文件的标志 ID
name	Varchar (45)		上传文件名称
Content	Blob		上传文件字节流

运行结果

应用部署成功后，在浏览器的地址栏输入“http://localhost:8080/MyUpload/commonFileupload/index.html”，将得到如图 5-20 所示的运行结果。



图 5-20 文件上传界面

为了演示多文件上传，页面提供了三个上传文件输入框。单击【浏览】按钮，选择要上传的文件。单击【开始上传】按钮，文件开始上传。最后得到如图 5-21 所示的运行结果，返回页面显示上传的文件名称和大小。到指定的上传目录，可以看到刚才上传的文件。

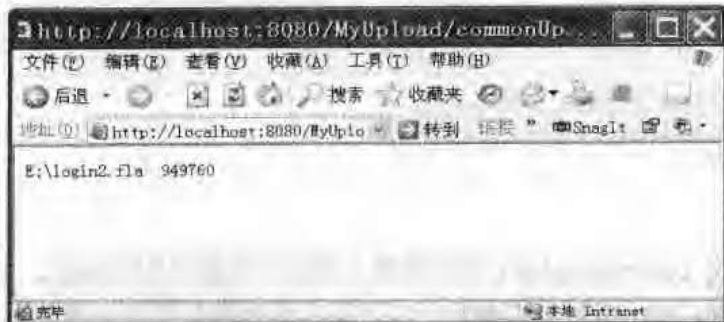


图 5-21 文件上传成功提示页面

在浏览器的地址栏输入“<http://localhost:8080/MyUpload/commonFileupload/index2.html>”，将得到如图 5-22 所示的运行结果。



图 5-22 上传文件选择界面

单击【开始上传】按钮上传文件。最后返回的页面显示上传文件的大小和名称。可以打开数据库查看数据的上传结果。

讨论与思考

文件上传的实质，就是在服务器与客户端之间建立起一条文件流通道，将文件信息流传递到服务器上。可以使用基本的文件流操作来实现，如本例中的解决方案 1；也可以采用一些免费的开源组件如 jspSmartUpload、common-upload 等。

但是上述方案仅能解决较小的、批量不大的文件上传问题，如果要想实现复杂的文件上传，如断点续传等，就需要在客户端实现更为复杂的操作，此时，Applet 将是一个不错的选择。有兴趣的读者可以参阅相关的资料。

知识点索引

文件上传；流操作；jspSmartUpload；common-upload。

例程 5-5：在 Web 应用中控制文件下载

目的

- (1) 文件流操作；

- (2) 访问服务器端的文件资源;
- (3) 资源请求转发。

问题

在 Web 应用系统中,经常需要发布文件资源供客户端下载。通常采用的方式是将要下载的文件放在一个统一的目录,如download下。如何实现文件资源的发布,既能确保资源的高效访问,又能实现对发布资源的控制,防止资源被非法访问或被盗链?

解决方案 1: 利用文件流操作实现文件下载

利用 Servlet 将下载的文件生成文件输出流,输出到客户端。客户下载前必须通过登录界面登录,系统将登录信息放入 session。Servlet 在将文件输出到客户端前,检查 session 中的值来判断是否是合法的访问,从而实现对下载的控制,防止盗链现象。

实现步骤

- (1) 生成登录界面页面 index.jsp 用来获取用户信息;
- (2) 生成登录信息处理页面 login.jsp 将用户信息添加到 session;
- (3) 根据下载目录的内容,生成包含下载内容的列表 downlist.jsp;
- (4) 通过 Servlet 将下载文件生成输出流输出到客户端。

示例代码

本例程的所有代码都位于 Netbeans 的项目文件夹 MyDownload 下。

程序 5-21: index.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>下载登录</title>
</head>
<body>

<table cellSpacing=1 cellPadding=2 width=300 bgColor=#e6e6e6 border=0>
  <form action="login.jsp" method=post target=_top>
    <tbody>
      <tr>
        <td width="100%" bgColor=#3face4 height=26>&nbsp;<b><font
color=#ffffff>用户登录</font></b></td></tr>
      <tr>
        <td width="100%" bgColor=#ffffff>
          <table cellSpacing=0 cellPadding=2 width="100%" border=0>
```


19

```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>下载文件列表</title>
  </head>
  <body>

    <h1>下载文件列表</h1>
    <%
String downPath=application.getInitParameter("DownLoadPath");
if(downPath==null|downPath.equals(""))
    out.println("下载路径设置错误,.....");
else{
    File fl=new File(downPath);
    File[] flist=fl.listFiles();
    for (int i=0;i<flist.length;i++){
    %>
    <ul>
      <a href=<%= "Download?name="+flist[i].getName() %>> <%=flist[i].getName() %>
      </a><br>
    </ul>
    <%
    }
    }
    %>
  </body>
</html>

```

程序说明：程序主要实现将下载目录中存放的所有文件通过链接列表的形式提供出来，为文件下载提供操作界面。为防止硬编码，将下载文件的目录作为 Web 上下文参数存放在 Web 应用配置文件 Web.xml 中，因此，首先调用 application 的 getInitParameter() 方法获取下载路径信息。然后，根据路径信息生成一个 File 实例，调用 File 实例的 listFiles() 方法即可获得下载路径下的所有文件。最后，将下载文件通过列表的形式发布到页面，并为每个文件提供一个下载链接地址。下载链接地址为负责下载文件的 Servlet 的 URL 地址，利用 URL 重写的方式将文件名称作为参数附在链接地址后面，传递到下载 Servlet 组件。

程序 5-24: Download.java

```

package com.download;
import java.io.*;
import java.net.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Download extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    //进行 session 检查，确认用户是否登录
    String ID=(String)request.getSession().getAttribute("UserID");
    if(ID==null){

```



```

        response.setCharacterEncoding("UTF-8");
        PrintWriter p=response.getWriter();

        p.println(" Access denied.....");
        return;
    }
    javax.servlet.ServletOutputStream out = response.getOutputStream();

    String path= getServletConfig().getServletContext().
getInitParameter("DownloadPath");
    String fileSep = System.getProperty("file.separator");
    String name=request.getParameter("name");
    java.io.File file = new java.io.File(path + fileSep+name);
    if (!file.exists()) {
        System.out.println(file.getAbsolutePath() + " 文件不存在!");
        return;
    }
    // 读取文件流
    java.io.FileInputStream fileInputStream = new java.io.FileInputStream(file);
    // 下载文件
    // 设置响应头和下载保存的文件名
    if (name != null && name.length() > 0) {
        response.setContentType("application/x-msdownload");
        response.setHeader("Content-Disposition", "attachment; filename=\"" + name + "\"");
        if (fileInputStream != null) {
            int filelen = fileInputStream.available();
            //文件太大时内存不能一次读出, 要循环
            byte a[] = new byte[filelen];
            fileInputStream.read(a);
            out.write(a);
        }
        fileInputStream.close();
        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        processRequest(request, response);
    }

    public String getServletInfo() {
        return "Short description";
    }
}

```

程序说明: 程序实现文件下载的核心功能。下载文件的功能代码实现主要在方法 `processRequest()` 中。

首先检查 Session 中的 UserID 属性,判断用户是否登录,来实现对下载的控制,避免非法盗链。下载文件操作的步骤如下:

(1) 获取下载文件的完整路径名称。首先调用 `getServletConfig().getServletContext().getInitParameter("DownloadPath")` 来获取存储在配置文件中的下载路径;然后,调用 `System.getProperty("file.separator")` 获取服务器文件系统的分隔符;最后调用 `request` 的 `getParameter("name")` 方法获取客户请求下载的文件名称。将上述三种信息组合便形成要下载文件的完整路径。

(2) 根据文件名称创建 File 文件,进而创建一个 `FileInputStream` 对象。

(3) 根据 `response` 对象创建一个 `ServletOutputStream` 对象。

(4) 调用 `setContentType("application/x-msdownload")` 设置 `response` 对象的内容类型。

(5) 调用 `setHeader("Content-Disposition", "attachment; filename=\"" + name + "\"")` 设置 `response` 对象的头部信息。

(6) 读取 `FileInputStream` 对象中的内容,并写入 `ServletOutputStream` 对象。

运行准备

将下载文件路径作为 Web 上下文参数添加到配置文件 `Web.xml`,添加的内容如下:

```
<context-param>
    <description>下载文件路径</description>
    <param-name>DownloadPath</param-name>
    <param-value>E:\001</param-value>
</context-param>
```

运行结果

部署完毕后,启动应用程序,在浏览器的地址栏中输入“`http://localhost:8080/MyDownload/index.jsp`”,将出现如图 5-23 所示的运行结果。

这里先不登录,先来验证以下程序是否能够防止盗链和非法访问。在浏览器的地址栏中输入 `http://localhost:8080/MyDownload/downlist.jsp`,则得到如图 5-24 所示的下载文件列表。



图 5-23 应用登录界面

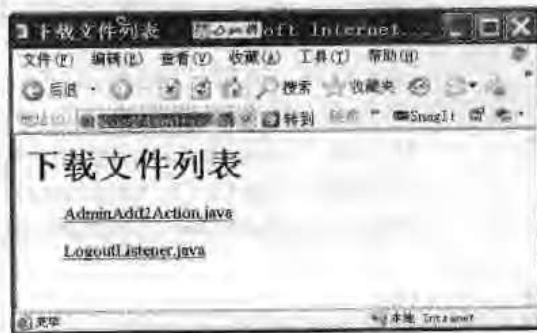


图 5-24 下载文件列表

单击其中的任意一个下载文件,将得到如图 5-25 所示的运行结果。可以看到,由于没有登录,下载文件的请求被禁止。

在浏览器的地址栏中输入“`http://localhost:8080/MyDownload/index.jsp`”,调出登录界面登录后,重新进入下载列表页面 `download.jsp`,选择任意文件下载,将得到如图 5-26 所示的下载界面。

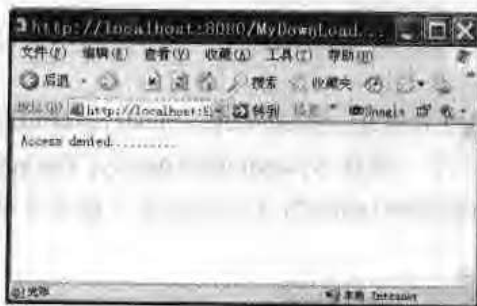


图 5-25 下载请求被拒绝提示信息



图 5-26 文件下载成功

单击【保存】按钮，选择合适的保存文件路径，则文件成功被下载。

解决方案 2：利用 RequestDispatcher 实现文件下载

设置 response 的内容类型为非 text/html 类型，设置 response 的头部为附件类型，获取要下载的文件对象，最后将请求和响应利用 forward 转向到 RequestDispatcher 对象。由于客户端浏览器发现响应的类型不是普通的 text/html 类型，则自动提示客户下载文件到客户端本地保存。

实现步骤

关于生成下载文件列表的方法和实现防盗链的方法与解决方案 1 完全一致，为节省篇幅，仅通过一个实现下载功能的 JSP 页面来演示如何利用 RequestDispatcher 实现文件下载。

示例代码

程序 5-25: download.jsp

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<%@ page import="java.net.URLEncoder" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%
    response.setContentType("application/x-download");//设置为下载
    String filenamedownload = "/download/下载文件.doc";//即将下载的文件的路径
    String filenamedisplay = "下载文件.doc";//下载文件时显示的文件保存名称
    filenamedisplay = URLEncoder.encode(filenamedisplay,"UTF-8");
    response.addHeader("Content-Disposition","attachment;filename=" + filenamedisplay);

    try
    {
        RequestDispatcher dispatcher = application.getRequestDispatcher
        (filenamedownload);
        if(dispatcher != null)
        {
            dispatcher.forward(request,response);
        }
        response.flushBuffer();
    }
}
```

```

    catch(Exception e)
    {
        e.printStackTrace();
    }
    finally
    {
    }

    }
}
%>

```

程序说明：在本示例程序中，为利用 RequestDispatcher 实现文件下载，采用以下基本步骤：

- (1) 调用 response 的 setContentType("application/x-download")方法设置响应内容类型；
- (2) 调用 URLEncoder 的 encode(filenamedisplay,"UTF-8")方法将文件名进行编码，防止在客户端显示的文件名称为乱码；
- (3) 调用 response 的 addHeader("Content-Disposition","attachment;filename="+ filenamedisplay) 方法设置响应的头部信息；
- (4) 获取要下载文件的 RequestDispatcher 对象；
- (5) 调用 RequestDispatcher 的 forward()方法将当前请求和响应转向下载文件；
- (6) 调用 response 的 flushBuffer()方法将响应内容输出到客户端。

运行准备

为演示程序运行，在 Web 应用下面建立一个新的目录 download，并在下面放置一个名为“下载文件”的 Word 文档。

运行结果

部署完毕后，启动应用程序，在地址栏中输入 <http://localhost:8080/MyDownLoad/download.jsp>，将出现如图 5-27 所示的运行结果。单击【保存】按钮可将文件成功保存在客户端。

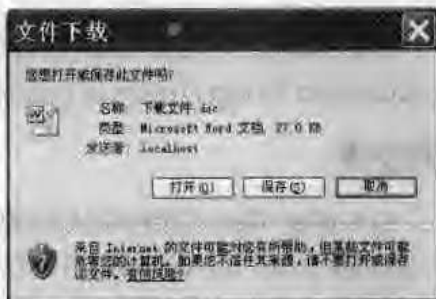


图 5-27 文件下载信息

讨论与思考

实现文件下载，最简单的方式就是直接提供一个到文件下载位置的链接，如 `下载 a.rar`，这样，当用户单击链接的时候就会弹出 IE 的【保存文件】对话框，然后下载。但是这样做的缺点很明显：

(1) 浏览器有时候对于一些常见的文件类型，如 Word 文档 (.doc 文件)，会调用相关的应用程序打开，而不是提示客户保存。

(2) 对下载文件完全失去控制，最大的隐患就是盗链的问题。一个用户可以直接通过 URL 地址随意地引用这个文件，造成文件的盗用，而且也无法实现对用户下载权限、线程数量、带宽等高级控制管理。

本例提供了两种解决方案，第一种基于文件流的方式，第二种基于 RequestDispatcher 方式，它们都很好地解决了防盗链问题。相对于基于 RequestDispatcher 方式，基于文件流的方式功能更强大，主要表现在：

(1) RequestDispatcher 只能转向 Web 应用内的资源文件，这就要求下载文件必须存放在 Web 应用程序内的路径下，而基于文件流的下载方式，文件可以存放在任意位置而不受限制。

(2) RequestDispatcher 只是将请求响应转向 Web 应用内的资源文件，具体的文件在服务器和浏览

器之间的传输,是由服务器来决定的,不受应用程序的控制;基于文件流的方式,开发人员可以控制文件流的传输方式、线程数量等,功能强大得多。

知识点索引

下载;流操作;RequestDispatcher;设置响应头部信息。

例程 5-6: 为 Web 应用添加日志功能

目的

- (1) 如何利用 Web 上下文监听器实现对 Web 应用状态变化的监听。
- (2) 演示如何为应用程序添加日志功能。

问题

B/S 模式下的服务器端的程序往往是每周 24×7 小时连续运行在服务器上,不间断地为客户提供服务,那么监控服务器端应用程序的运行状态监控就变得十分重要。如何通过日志功能来自动记录服务器端应用程序的开始运行时间和终止时间呢?

解决方案 1: 利用服务器自身的日志功能

利用 Web 上下文监听器实现对 Web 应用状态变化的监听,在监听器中利用 Web 上下文对象 ServletContext 的 log()方法实现日志功能,来记录 Web 应用程序状态的变化。

实现步骤

- (1) 创建 Web 应用程序侦听程序 ContextLogger, 并实现 Web 上下文监听接口。

具体操作为:在 netbeans 开发环境下,选中项目文件路径 log,单击右键,在弹出的快捷菜单中选择【新建】→【Web 应用程序侦听程序】,弹出【新建 Web 应用程序侦听程序】对话框,如图 5-28 所示。

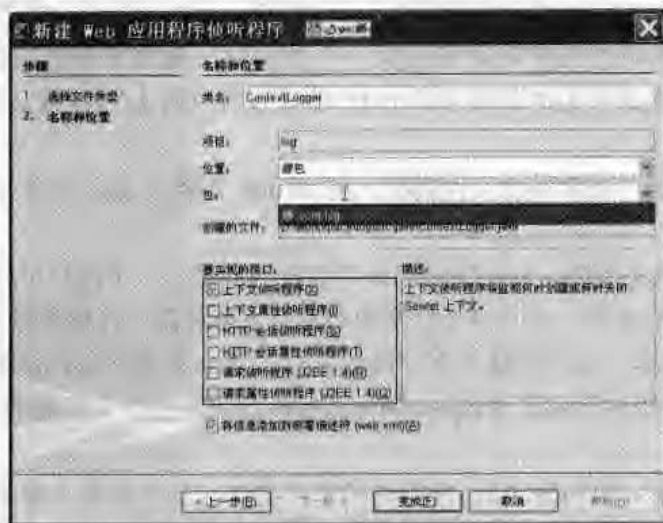


图 5-28 新建 Web 应用程序侦听程序

注意，要在【要实现接口】列表选中【上下文侦听程序】，以便监听器实现 Web 上下文侦听。单击【完成】，则 Web 应用程序侦听程序成功创建。

(2) 在 `contextInitialized()` 方法中，调用 `ServletContextEvent` 对象的 `getServletContext()` 获取 Web 上下文对象。

(3) 调用 Web 上下文对象的 `log()` 方法来实现日志功能，来记录应用的载入时间。

(4) 在 `contextDestroyed()` 方法中，重复步骤 (2) 和 (3) 的操作实现 Web 应用关闭时间的记录。

示例代码

本例程的所有代码保存在 netbeans 的工程 log 下。

程序 5-26: ContextLogger.java

```
package com.log;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextListener;
import javax.servlet.ServletContextEvent;
public class ContextLogger implements ServletContextListener {
    public void contextInitialized(ServletContextEvent evt) {
        ServletContext context = evt.getServletContext();
        context.log(new java.util.Date()+"Web 应用被载入.....");
    }
    public void contextDestroyed(ServletContextEvent evt) {
        ServletContext context = evt.getServletContext();
        context.log(new java.util.Date()+"Web 应用被关闭.....");
    }
}
```

运行结果

打开应用服务器的日志文件（不同的应用服务器，日志文件的存放路径不一致，具体位置查看相关资料。如果使用 Tomcat，则日志文件位于 Tomcat 安装目录下的 logs 子目录下），将看到如下内容：

2007-05-16 09:38:26 StandardContext[/log]Wed May 16 09:38:26 CST 2007:Web 应用被载入.....

解决方案 2：利用 log4j 实现日志功能

log4j 是一个开源的项目，用于帮助应用程序实现强大的日志功能。利用 log4j 为应用添加日志功能。

知识点链接

使用 log4j 主要涉及以下 3 个基本概念。

公共类 `Logger`：负责处理日志记录的大部分操作。

公共接口 `Appender`：负责控制日志记录操作的输出。

公共抽象类 `Layout`：负责格式化 `Appender` 的输出。

(1) Logger

日志记录器 (`Logger`) 是日志处理的核心组件。log4j 具有以下 5 种正常级别 (Level)。

static Level DEBUG: DEBUG 级别指出细粒度信息事件, 对调试应用程序是非常有帮助的。

static Level INFO: INFO 级别表明消息在粗粒度级别上突出强调应用程序的运行过程。

static Level WARN: WARN 级别表明会出现潜在错误的情形。

static Level ERROR: ERROR 级别指出虽然发生错误事件, 但仍然不影响系统的继续运行。

static Level FATAL: FATAL 级别指出每个严重的错误事件将会导致应用程序的退出。

另外, 还有以下两个可用的特别的日志记录级别。

static Level ALL: ALL 级别是最低等级的, 用于打开所有日志记录。

static Level OFF: OFF 级别是最高等级的, 用于关闭所有日志记录。

Logger 将只输出那些级别高于或等于它的级别的信息。如果没有设置 Logger 的级别, 那么它将会继承最近的祖先的级别。

有很多方法可以创建一个 Logger, 下面方法可以取回 root 日志记录器:

```
Logger logger = Logger.getRootLogger();
```

还可以这样创建一个新的日志记录器:

```
Logger logger = Logger.getLogger("MyLogger");
```

比较常用的用法, 就是根据类名实例化一个静态的全局日志记录器:

```
static Logger logger = Logger.getLogger(test.class);
```

所有这些创建的叫 “logger” 的日志记录器都可以用下面方法设置级别:

```
logger.setLevel((Level)Level.WARN);
```

可以使用 7 个级别中的任何一个进行创建。

(2) Appender

Appender 控制日志怎样输出。下面列出一些可用的 Appender。

ConsoleAppender: 使用用户指定的布局(layout) 输出日志事件到 System.out 或者 System.err。默认的目标是 System.out。

DailyRollingFileAppender: 扩展 FileAppender, 因此多个日志文件可以以一个用户选定的频率进行循环日志记录。

FileAppender: 把日志事件写入一个文件

RollingFileAppender: 扩展 FileAppender 备份容量达到一定大小的日志文件。

还可以继承 Appender 接口, 创建以定制的方式进行日志输出的 Appender。

(3) Layout

Appender 必须使用一个与之相关联的 Layout, 这样它才能知道怎样格式化它的输出。当前, log4j 具有以下 3 种类型的 Layout。

HTMLLayout: 格式化日志输出为 HTML 表格。

PatternLayout: 根据指定的转换模式格式化日志输出, 或者如果没有指定任何转换模式, 就使用默认的转换模式。

SimpleLayout: 以一种非常简单的方式格式化日志输出, 它打印级别 Level, 然后跟着一个破折号 “——”, 最后才是日志消息。

实现步骤

(1) 将 log4j 组件的 Jar 添加到项目的 classpath 下;

(2) 编写 log4j 的初始化配置文件 log4j.properties, 并把它放置在项目的源文件夹下, 这样就能够确保 log4j 在初始化的时候能够成功加载;

(3) 创建 Web 应用程序侦听程序 ContextLogger2, 并实现 Web 上下文侦听接口;

(4) 为避免硬编码, 在 Web.xml 文件中添加 log4j 的配置文件的名称 log4j.properties 作为上下文参数, 代码如下:

```
<context-param>
    <description>log4j 组件的初始化配置文件</description>
    <param-name>logger-config</param-name>
    <param-value>log4j.properties</param-value>
</context-param>
```

(5) 在程序 ContextLogger2 的方法利用 log4j 实现日志功能。

示例代码

程序 5-27: ContextLogger2.java

```
package com.log;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextListener;
import javax.servlet.ServletContextEvent;
import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

public class ContextLogger2 implements ServletContextListener {
    private Logger log;

    public void contextInitialized(ServletContextEvent evt) {
        ServletContext context = evt.getServletContext();
        String realPath = context.getRealPath("/");
        String fileSep = System.getProperty("file.separator");
        if (realPath != null && (! realPath.endsWith(fileSep))) {
            realPath = realPath + fileSep;
        }
        PropertyConfigurator.configure(realPath +
            "WEB-INF/classes/" + context.getInitParameter("logger-config"));
        log = Logger.getLogger(ContextLogger2.class);
        String name = context.getServletContextName();
        log.info("Web 应用程序准备就绪: " + (name == null ? "" : name));
    }

    public void contextDestroyed(ServletContextEvent evt) {
        String name = evt.getServletContext().getServletContextName();
        log.info("Web 应用程序被关闭: " + (name == null ? "" : name));
    }
}
```

程序 5-28: log4j.properties

```
log4j.rootLogger=DEBUG, cons
log4j.logger.com.log=, myAppender

log4j.appender.cons=org.apache.log4j.ConsoleAppender

#configure the 'myAppender' appender
```



```

log4j.appender.myAppender=org.apache.log4j.RollingFileAppender
log4j.appender.myAppender.File=e:web.log
log4j.appender.myAppender.MaxBackupIndex=1
log4j.appender.myAppender.MaxFileSize=1MB

log4j.appender.cons.layout=org.apache.log4j.SimpleLayout
log4j.appender.myAppender.layout=org.apache.log4j.PatternLayout
log4j.appender.myAppender.layout.ConversionPattern=%-5p Logger:%c{1} Date: %d{ISO8601}
- %m%n

```

程序说明：log4j 组件的配置文件。在配置文件中，定义了日志信息的两种输出方式。第一种是输出到控制台的 ConsoleAppender，第二种是输出到指定文件的 myAppender。关于配置文件的详细定义，请参阅相关资料。

运行准备

为了正确输出日志到指定的日志文件，必须在运行前创建配置文件中指定的日志输出文件（在本例中为 e:web.log）。

运行结果

部署完毕后，启动应用程序，在【输出】窗口的【Tomcat 5.0】标签视图下，将看到输出到控制台的日志信息输出，如图 5-29 所示。

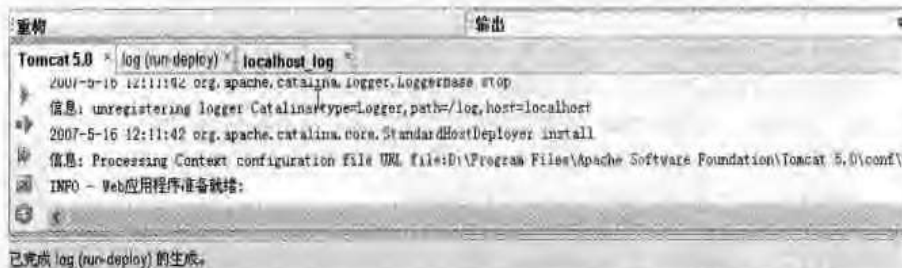


图 5-29 log4j 的输出信息

打开在 log4j.properties 定义的日志文件，（在本例为 e:web.log），同样也会看到输出的日志信息。

修改 log4j.properties 文件中的日志监控级别为 WARN，即将 log4j.rootLogger=DEBUG, cons

修改为 log4j.rootLogger=WARN, cons。

重新发布运行，在服务器的控制台和日志文件中将不会有日志输出。因为程序中日志语句的级别 INFO 低于配置文件中定义的日志输出级别 WARN。怎么样，体会到 log4j 强大的控制能力了吧？

讨论与思考

日志是企业开发中几乎必不可少的组成部分。在开发的过程中，它起着辅助调试的作用，可以帮助开发人员了解程序运行过程中某些变量的信息。在程序的运行阶段，它可以帮助系统维护人员监控系统的行为和运行状态，确保系统的运行安全。

这里展示了 Web 应用程序的两种解决方案。基于服务器的解决方案，使用应用服务器自身的日志功能，操作步骤简单，但功能也相对单一；基于 log4j 的解决方案，日志功能相对要强大得多。log4j 的好处在于：

(1) 通过修改配置文件, 就可以决定 log 信息输出到何处 (console, 文件,...), 是否输出。这样, 在系统开发阶段可以打印详细的 log 信息以跟踪系统运行情况而在系统稳定后可以关闭 log 输出, 从而在能跟踪系统运行情况的同时, 减少了垃圾代码 (System.out.println(...)等)。

(2) 使用 log4j, 实现整个 Web 系统的统一的 log 机制, 有利于系统的整体规划和控制。

知识点索引

日志; Web 上下文; 监听器; log4j。

本章小结

在 Web 应用中, 诸如上传、下载、日志、图表等功能的实现, 对整个 Web 应用及客户需求的满足都具有重要的影响。由于上述功能使用的频率比较高, 因此在一定程度上又会影响到应用程序的性能。通过以上示例的演示, 相信开发人员对如何解决上述问题, 都有了深入的认识。在实际的项目开发实践中, 要针对项目的具体问题, 通过认真分析, 确定最适合的解决方案。

第6章 应用架构

如果我看得更远的话，那是因为我站在巨人的肩膀上

——艾萨克·牛顿

当比较全面地学习了 Java EE 的各项开发技术，掌握了各种编程技巧后，自然就要跃跃欲试地开始开发一个 Java EE 的软件系统了。这时，有两种方法：一是白手起家，从系统的规划、设计到代码实现都全部自己动手实现；二是在充分分析系统需求的基础上，参照已有的成熟应用框架，来架构开发自己的软件系统。对于上述两种开发方式，往往有着两种截然不同的结局。对于前者，可能由于前期系统设计上的疏忽和缺陷导致系统开发被迫停止，回头重新设计系统的架构，除了白白浪费大量时间精力外，还必须面对项目的延期交付、开支预算的超出。对于后者，基于成熟的架构来开发，项目的整个进度将是顺利地进行，一切都是那么的水到渠成。

造成上述两种截然不同结果的关键就在于软件架构的应用。聪明的学者总是会善于利用前人的成果，在架构 Java EE 应用系统时，也毫不例外。由于 Java 是一种开源的软件开发语言，世界各地的广大 Java 开发人员贡献出了大量的应用开发框架，因此应用系统的开发人员可以基于它们来架构自己的应用。

本文将通过完整的示例演示如何基于成熟的架构来开发 Web 应用。

预备知识：软件架构基础

1. 什么是软件架构

软件架构 (Software Architecture) 是一系列相关的抽象模式，用于指导大型软件系统各个方面的设计。软件架构是一个系统的草图，其描述的对象是直接构成系统的抽象组件。各个组件之间的连接则明确和相对细致地描述了组件之间的通信。在实现阶段，这些抽象组件被细化为实际的组件，比如具体某个类或者对象。在面向对象领域中，组件之间的连接通常用接口来实现。

2. 架构的目标是什么

正如同软件本身有其要达到的目标一样，架构设计也有其要达到的目标。一般而言，软件架构设计要达到如下的目标。

可靠性 (Reliable)：软件系统对于用户的商业经营和管理来说极为重要，因此软件系统必须非常可靠。

安全性 (Secure)：软件系统所承担的交易的商业价值极高，系统的安全性非常重要。

可扩展性 (Scalable)：软件必须能够在用户的使用率、用户的数目增加很快的情况下，保持合理的性能。只有这样，才能适应用户的市场扩展。

可定制化 (Customizable)：同样的一套软件，可以根据客户群的不同和市场需求的变化进行调整。

可扩展性 (Extensible)：在新技术出现的时候，一个软件系统应当允许导入新技术，从而对现有系统进行功能和性能的扩展。

可维护性 (Maintainable)：软件系统的维护包括两方面，一是排除现有的错误，二是将新的软件需求反映到现有系统中去。一个易于维护的系统可以有效地降低技术支持的花费。

用户体验：软件系统必须易于使用。

市场时机：软件用户要面临同业竞争，软件提供商也要面临同业竞争。以最快的速度争夺市场先机非常重要。

3. 架构的种类

根据关注的角度不同,可以将架构分成以下3种。

逻辑架构:软件系统中元素之间的关系,比如用户界面、数据库、外部系统接口、商业逻辑元素,等等。

物理架构:软件元素是怎样部署到硬件上的。

系统架构:系统的非功能性特征,如可扩展性、可靠性、强壮性、灵活性、性能等。

此外,从每一个角度上看,都可以看到架构的两要素:元素划分和设计决定。

首先,一个软件系统中的元素首先是逻辑元素。这些逻辑元素如何放到硬件上,以及这些元素如何为整个系统的可扩展性、可靠性、强壮性、灵活性、性能等做出贡献,是非常重要的信息。

其次,进行软件设计需要做出的决定中,必然会包括逻辑结构、物理结构,以及它们如何影响到系统的所有非功能性特征。在这些决定中,很多决定一旦作出,是很难更改的。

4. 框架 (Framework)

伴随着软件开发的发展,在多层的软件开发项目中,可重用、易扩展、而且是经过良好测试的软件组件,越来越为人们所青睐。这意味着人们可以将充裕的时间用在分析、构建业务逻辑的应用上,而非繁杂的代码工程上。于是人们将相同类型问题的解决途径进行抽象,抽取成一个应用框架。这也就是我们所说的框架。从这个意义上说,框架可以看作架构的结果和产物。

框架的体系提供了一套明确机制,从而让开发人员很容易地扩展和控制整个框架开发上的结构。

对于常见的 Web 应用框架,一般具有以下明显的层次结构。

表示逻辑层:用来实现信息的展示和交互。

业务逻辑层:用来实现具体的业务逻辑。这也是应用开发中最容易发生变更的需求。

持久存储层:用来访问和操作存储在关系数据库、目录服务器等持久化系统中的信息。

例程 6-1: 利用 EJB 实现公告发布系统

目的

- (1) 演示如何创建 EJB 组件;
- (2) 演示如何在 Web 应用程序中调用 EJB 组件。

问题

如何利用 EJB 组件实现公告发布系统,并通过 Web 组件与用户进行交互?

解决方案

利用实体 EJB 来抽象存储在关系数据库中的公告信息,利用会话 EJB 实现对实体 EJB 的访问代理,在 Servlet 组件中调用会话 EJB 来访问代表公告信息的实体 EJB,通过 JMS 消息服务与消息驱动 EJB 实现松散耦合,通过消息驱动 EJB 调用持久化管理器来创建新的实体 EJB,实现公告信息的发布。

实现步骤

1. 建立企业应用程序

为演示 Web 组件对 EJB 组件的调用过程,创建的企业应用程序中必须既包含一个 EJB 模块,又包含一个 Web 应用模块。还需要目标服务器支持 EJB 规范。因此,本例选择目标应用服务器 Sun Java System Application Server。

2. 创建持久性单元

实体 EJB 基于容器提供的持久化服务来完成持久化操作。因此，在创建 EJB 前，首先创建一个持久性单元。

说明：持久化服务是 EJB 容器提供的一种抽象，它在底层也是通过在例程 4-2 中所示的各种技术来实现的，因此必须要配置完整的数据源信息及指定实现持久化的方法。

具体操作如下：

(1) 右键单击 EJB 模块，然后从弹出的快捷菜单中选择【新建】→【文件/文件夹】，打开【新建文件】对话框；

(2) 从【类别】中选择【持久性】，从【文件类型】中选择【持久性单元】，如图 6-1 所示，然后单击【下一步】；



图 6-1 创建持久性单元

(3) 保留默认的持久性单元名称；

(4) 在【持久性提供程序】中选择【TopLink（缺省）】；

(5) 在【数据源】中选择缺省的数据源【jdbc/sample】；

(6) 检查是否为持久性单元选中了【使用 Java 事务 API】，以及【表生成策略】是否设置为【创建】，以便在部署应用程序时创建基于实体类的表，如图 6-2 所示；



图 6-2 设置持久性单元属性

(7) 单击【完成】。完成持久性单元的创建。

3. 创建实体 EJB notice, 用来代表公告信息

分别添加四个字段用来代表公告信息的属性。(详细代码如程序 6-1 所示)

4. 创建会话实体 Bean NoticeFacade, 作为实体 Bean 的访问代理

由于会话 Bean 与实体 Bean 都部署在同一个服务器上, 因此创建 EJB 的本地接口, 通过本地接口来访问实体 Bean。

5. 创建消息驱动 EJB NewNotice

NewNotice 根据消息实体中传递来的信息, 调用持久化管理器提供的服务, 将实体 EJB 保存到关系性数据库中。

EJB 3.0 规范允许开发人员使用标记将资源直接引入类中。打开源代码文件, 可以看到以下标注:

```
@MessageDriven(mappedName = "jms/NewMessage")
```

此标注向容器说明: 该组件是一个消息驱动 Bean, 并且该 Bean 使用 JMS 资源。当 IDE 生成类时, 将从类 (NewMessage.java) 名称派生资源的映射名 (jms/NewMessage)。JMS 资源被映射到目标的 JNDI 名称, Bean 从该目标中接收消息。“新建消息驱动 Bean”向导已为开发人员创建了 JMS 资源。通过 EJB 3.0 API, 可以从 Bean 类中查找 JNDI 名称空间中的对象, 这样就不需要配置部署描述符来指定 JMS 资源了。

下面使用标注将 MessageDrivenContext 资源引入类中, 然后注入 PersistenceContext 资源, EntityManager API 将使用该资源来管理持久性实体实例。

6. 创建 Servlet NoticeList 显示公告信息列表

在 Servlet 中调用会话 EJB。具体操作如下。

(1) 引入会话 EJB。在源代码编辑器中, 单击右键, 在弹出的快捷菜单中选择【企业资源】→【调用 Enterprise Bean】, 在弹出的对话框中选择【NoticeFacade】, 则将会话 Bean 引入到 Web 组件的上下文中。在代码源文件中也可以看到如下的标注:

```
@EJB
private NoticeFacadeLocal noticeFacade;
```

(2) 调用 noticeFacade 的 findAll() 方法获取公告信息列表。

7. 创建 Servlet PostNotice 来发布新的公告信息

具体操作如下:

(1) 通过标注引入 JMS 消息资源;

(2) 创建到 JMS 消息队列的链接;

(3) 创建消息并发送到队列中。

示例代码

本例程的所有代码保存在 netbeans 的工程 notice 下。

程序 6-1: Notice.java

```
package com.hyl;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
```



```

import javax.persistence.Id;

/**
 * 实体类 Notice
 *
 * @author hyl
 */
@Entity
public class Notice implements Serializable {
    private String title;
    private String body;
    private java.sql.Timestamp publishDate;
    private String author;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    /** Creates a new instance of Notice */
    public Notice() {
    }

    /**
     * 获取此 Notice 的 id。
     * @return id
     */
    public Long getId() {
        return this.id;
    }

    /**
     * 将此 Notice 的 id 设置为指定的值。
     * @param id, 新建 id
     */
    public void setId(Long id) {
        this.id = id;
    }

    /**
     * 返回对象的散列代码值。该实现根据此对象
     * 中 id 字段计算散列代码值。
     * @return 此对象的散列代码值。
     */
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (this.id != null ? this.id.hashCode() : 0);
        return hash;
    }

    /**
     * 确定其他对象是否等于此 Notice。当且仅当
     * 参数不为 null 且该参数是具有与此对象相同 id 字段值的 Notice 对象时，

```

```

    * 结果才为 <code>true</code>。
    * @param 对象，要比较的引用对象
    * 如果此对象与参数相同，则 @return <code>true</code>;
    * 否则为 <code>false</code>。
    */
    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Notice)) {
            return false;
        }
        Notice other = (Notice) object;
        if (this.id != other.id && (this.id == null || !this.id.equals(other.id))) return false;
        return true;
    }

    /**
     * 返回对象的字符串表示法。该实现根据 id 字段
     * 构造此表示法。
     * @return 对象的字符串表示法。
     */
    @Override
    public String toString() {
        return "com.hyl.Notice[id=" + id + "]";
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getBody() {
        return body;
    }

    public void setBody(String body) {
        this.body = body;
    }

    public String getAuthor() {
        return author;
    }

    public void setAuthor(String author) {
        this.author = author;
    }

    public java.sql.Timestamp getPublishDate() {
        return publishDate;
    }

    public void setPublishDate(java.sql.Timestamp publishDate) {
        this.publishDate = publishDate;
    }
}

```


程序说明：作为一个实体 Bean，程序用来代表公告信息。在 EJB 3.0 以后，实体 EJB 的开发变得十分便捷。从上面的代码可以看出，除了使用几个特殊的标注信息外，代码与一个普通的 JavaBean 几乎没什么区别。

程序 6-2: NoticeFacade.java

```
package com.hyl;

import java.util.List;
import javax.ejb.Stateless;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**
 *
 * @author hyl
 */
@Stateless
public class NoticeFacade implements NoticeFacadeLocal {

    @PersistenceContext
    private EntityManager em;

    /** Creates a new instance of NoticeFacade */
    public NoticeFacade() {
    }

    public void create(Notice notice) {
        em.persist(notice);
    }

    public void edit(Notice notice) {
        em.merge(notice);
    }

    public void destroy(Notice notice) {
        em.merge(notice);
        em.remove(notice);
    }

    public Notice find(Object pk) {
        return (Notice) em.find(Notice.class, pk);
    }

    public List findAll() {
        return em.createQuery("select object(o) from Notice as o").getResultList();
    }
}
```

程序说明：作为会话 Bean，用来实现对实体 Bean 的访问代理。

程序 6-3: NewNotice.java

```
package com.hyl;
```

```

import javax.annotation.Resource;
import javax.ejb.ActivationConfigProperty;
import javax.ejb.MessageDriven;
import javax.ejb.MessageDrivenContext;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.ObjectMessage;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

/**
 * 实体类 NewNotice
 *
 * @author hyl
 */
@MessageDriven(mappedName = "jms/NewNotice", activationConfig = {
    @ActivationConfigProperty(propertyName = "acknowledgeMode", propertyValue =
        "Auto-acknowledge"),
    @ActivationConfigProperty(propertyName = "destinationType", propertyValue =
        "javax.jms.Queue")
})
public class NewNotice implements MessageListener {

    @PersistenceContext
    private EntityManager em;
    @Resource
    private MessageDrivenContext mdc;
    /** Creates a new instance of NewNotice */
    public NewNotice() {
    }

    public void onMessage(Message message) {
        ObjectMessage msg = null;
        try {
            if (message instanceof ObjectMessage) {
                msg = (ObjectMessage) message;
                Notice e = (Notice) msg.getObject();
                persist(e);
            }
        } catch (JMSException e) {
            e.printStackTrace();
            mdc.setRollbackOnly();
        } catch (Throwable te) {
            te.printStackTrace();
        }
    }

    public void persist(Object object) {

```



```

        // TODO:
        em.persist(object);
    }
}

```

程序说明：作为一个消息驱动 EJB，它将传递来的消息实体，通过持久化服务，实现持久化操作。

程序 6-4: NoticeList.java

```

package com.hyl.web;

import com.hyl.Notice;
import com.hyl.NoticeFacade;
import com.hyl.NoticeFacadeLocal;
import java.io.*;
import java.util.Iterator;
import java.util.List;
import javax.ejb.EJB;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 *
 * @author hyl
 * @version
 */
public class NoticeList extends HttpServlet {

    @EJB
    private NoticeFacadeLocal noticeFacade;

    /** Processes requests for both HTTP GET and POST methods.
     * @param request servlet request
     * @param response servlet response
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();

        out.println("<html>");
        out.println("<meta http-equiv='Content-Type' content='text/html'; charset='UTF-8'>");
        out.println("<head>");
        out.println("<title>公告栏</title>");
        out.println("</head>");
        out.println("<body>");
        List notices = noticeFacade.findAll();
    }
}

```

```

        for (Iterator it = notices.iterator(); it.hasNext();) {
            Notice elem = (Notice) it.next();
            out.println(" <b >" + elem.getTitle() + " <font color='red'> " + elem.
                getAuthor() + "</font> </b><br />");
            out.println(elem.getBody() + "<br /> ");
            java.sql.Timestamp pdate = elem.getPublishDate();
            out.println(pdate.toString() + "<br /> ");
            // out.println(elem.getAuthor() + "<br /> ");
        }
        out.println("<a href='PostNotice'>发布新的公告</a>");
        out.println("</body>");
        out.println("</html>");

        out.close();
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        processRequest(request, response);
    }

    /** Returns a short description of the servlet.
     */
    public String getServletInfo() {
        return "Short description";
    }

    // </editor-fold>
}

```

程序说明：程序主要用来显示公告信息列表。它通过标记在 Web 组件中引入会话 EJB 组件的引用，然后调用 EJB 的方法来获取代表公告信息的实体 Bean，最后将信息输出到页面。

程序 6-5: PostNotice.java

```

package com.hyl.web;

import com.hyl.Notice;
import java.io.*;
import javax.annotation.Resource;
import javax.jms.ConnectionFactory;
import javax.jms.JMSException;
import javax.jms.MessageProducer;
import javax.jms.ObjectMessage;
import javax.jms.Session;
import javax.jms.Connection;

```



```

import javax.jms.Queue;

import javax.servlet.*;
import javax.servlet.http.*;

public class PostNotice extends HttpServlet {
    @Resource(mappedName="jms/NewNoticeFactory")
    private ConnectionFactory connectionFactory;

    @Resource(mappedName="jms/NewNotice")
    private Queue queue;

    /** Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
     * @param request servlet request
     * @param response servlet response
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        response.setContentType("text/html;charset=UTF-8");

        // Add the following code to send the JMS message
        request.setCharacterEncoding("UTF-8");
        String title=request.getParameter("title");
        String body=request.getParameter("body");
        String author=request.getParameter("author");
        if(author==null)author="";
        if(author.equals(""))author="匿名";
        if ((title!=null) && (body!=null)) {
            try {
                Connection connection = connectionFactory.createConnection();
                Session session = connection.createSession(false, Session.AUTO_
                    ACKNOWLEDGE);
                MessageProducer messageProducer = session.createProducer(queue);

                ObjectMessage message = session.createObjectMessage();
                Notice e = new Notice();
                e.setTitle(title);
                e.setBody(body);
                e.setAuthor(author);
                e.setPublishDate(new
                    java.sql.Timestamp(System.currentTimeMillis()));
                message.setObject(e);
                messageProducer.send(message);
                messageProducer.close();
                connection.close();
                response.sendRedirect("NoticeList");

            } catch (JMSEException ex) {
                ex.printStackTrace();
            }
        }
    }
}

```

```

    }

    PrintWriter out = response.getWriter();

    out.println("<html>");
    out.println("<meta http-equiv='Content-Type' content='text/html'; charset='UTF-8'>");
    out.println("<head>");
    out.println("<title>公告发布栏</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<form>");
    out.println("作者: <input type='text' name='author'><br/>");
    out.println("标题: <input type='text' name='title'><br/>");
    out.println("内容: <textarea name='body'></textarea><br/>");
    out.println("<input type='submit' value='发布'><br/>");
    out.println("</form>");
    out.println("</body>");
    out.println("</html>");

    out.close();
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}

public String getServletInfo() {
    return "Short description";
}

// </editor-fold>
}

```

程序说明：程序用来实现公告信息的发布。它提供一个界面用来允许用户输入公告信息，并将用户提交的信息组装成实体 EJB，然后发送到消息对列中。

运行准备

根据 Java EE 5 的新特性，EJB 组件和 Web 组件所需的企业资源信息都会在发布时由应用服务器自动生成部署。

由于 Web 组件和 EJB 组件全部位于一个企业应用内，因此发布时，以企业应用 Notice 为单位进行发布。

应用部署完毕后，打开浏览器，在 IE 地址栏输入“http://localhost:4848/asadmin/index.html”进入管

理控制台，可以查看应用部署时自动生成的所需资源信息，如图 6-3 所示。

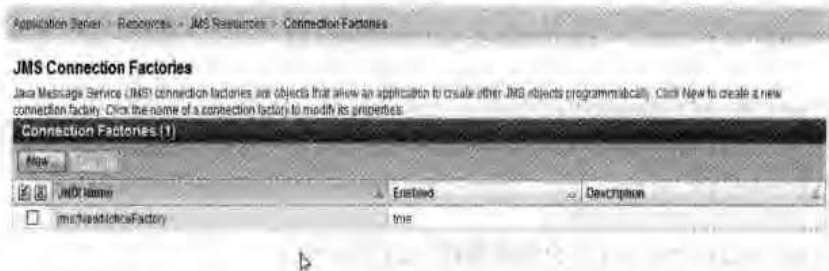


图 6-3 Web 应用发布后在服务器上自动创建的 JMS 资源

运行结果

应用部署成功后，启动应用，在浏览器的地址栏输入“[http://localhost:8080/Notice-war/ NoticeList](http://localhost:8080/Notice-war/NoticeList)”，将得到如图 6-4 所示的运行结果。



图 6-4 程序运行结果

由于尚未发布任何公告，因此公告列表为空。单击链接【发布新的公告】进入公告发布界面，如图 6-5 所示。



图 6-5 公告发布页面

输入要发布的消息后，单击按钮【发布】，重新回到公告列表页面。刷新页面，将得到如图 6-6 所示的运行结果。可以看到公告信息已经发布成功。



图 6-6 程序运行结果

讨论与思考

1. EJB 的特点

2006 年 6 月, EJB 3.0 规范正式发布。相对于以往的 EJB 规范, EJB 3.0 有了明显的改变。主要表现在以下几点。

(1) 元数据的引入: 以注释的形式表示元数据, 取代了过去大量的配置文件。

(2) 各种企业 Bean 的定义都只需要定义一些 POJO (Plain Old Java Objects, 简单旧式 Java 对象) 和 POJI (Plain Old Java Interfaces, 简单旧式 Java 接口) 配合元数据即可完成

(3) 回调机制 (Callback): 是用普通的方法加以一定的注释, 代替了原有的 ejbCreate 之类的方法。

(4) AOP 概念的引入: 允许在类定义中和其他类中对方法进行拦截。

(5) 依赖注入: 允许通过添加特定注释向属性注入所依赖的对象。

依赖注入不单是 EJB 3.0 规范的新特性, 而是整个 Java EE 5.0 规范的一个新特性。从本示例的实现可以看出, 在 Web 组件调用 EJB 变得那么简单。

基于 EJB 的 Java EE 应用中, 通过 EJB 对数据库的操作可以有两种方式: 实体 EJB 或者会话 EJB 中直接利用 JDBC 访问。在本示例中使用实体 EJB 来对数据库操作。

使用实体 EJB 来操作数据库有以下优点:

- ✎ 可以更好地贯彻 Java 面向对象的编程思想;
- ✎ 可以充分利用容器提供的事务、持久性等服务, 降低代码开发的难度, 尤其是在 EJB 3.0 版本以后, 这一优点更为明显;
- ✎ 移植性。

相对于利用实体 Bean 访问数据库, 通过 JDBC 直接访问数据库具有以下优点:

- ✎ 充分利用数据库能力, 比如数据库的缓存机制;
- ✎ 减少了对事务控制的资源;
- ✎ 利用自定义 SQL 可以按需要比较灵活地读取数据;
- ✎ 只需要一次数据查询, 减少了数据库操作。

但缺点也很明显:

- ✎ 与 Java EE 应用的面向对象设计相违背;
- ✎ 由于会话 EJB 代码中包含了自定义 SQL, 维护性差;
- ✎ 会话 EJB 中不得不包含 JDBC 的 API, 并且需要了解数据库结构。

2. EJB 设计模式

关于 EJB 的设计模式可以用一整本书来描述，但这里仅就本示例中使用的 EJB 设计模式进行简单论述。本示例中使用的设计模式主要有两个：会话代理模式和消息代理模式。

(1) 会话代理模式

通常项目实践中，客户端往往需要频繁地对服务器端数据进行操作。当采用实体 EJB 作为数据的抽象层时，如果直接让客户端程序与实体 EJB 交互，会产生仅一个业务需求便需要大量的 EJB 属性的操作。这直接导致如下问题：网络负载大（远程客户端时）、并发性能低、客户端与服务器端关联度大、可重用性和可维护性差等问题。

因此有必要在客户端与实体 EJB 层间加入会话 EJB 层，在会话 EJB 中实现商业逻辑并封装对实体 EJB 的操作。

会话代理模式的好处是：降低了网络负载，会话 EJB 可以调用实体 EJB 的本地接口；将商业逻辑与商业数据隔离；维护与开发方便；无状态的会话 EJB 还可以处理不同客户端的请求，显著提高性能。

会话代理模式因其简单使用，是目前使用很广的模式。但具体应用过程中应注意：避免将所有的操作封装到一个很大的会话 EJB 内；服务器端数据结构应由实体 EJB 实现，除非特例否则避免直接的数据库操作；会话 EJB 内某些系统通用操作的代码容易重复（比如权限检查等，解决办法是将系统通用服务封装在 Java Class 内）。

(2) 消息代理模式

如果一次客户请求需要操作多个 EJB 又不需要得到即时返回，对这种异步调用，通常应用消息代理模式。这种时候，如采用会话代理模式存在如下问题。

① 客户端等待返回的时间过长。一个会话 EJB 的实例在完成客户请求过程中涉及的每一次对其他实体 EJB 的调用过程中都会被锁定，直到得到实体 EJB 返回信息后才能进行下一步操作。这样造成客户不必要的等待，并很容易因时间导致整个事务失败。基于消息耦合的方式使得系统的响应时间大大缩短。

② 系统可靠性和容错性低。如果需要调用不同系统或服务器上或多个异构数据源的多个 EJB 时，任何一个环节出错，均导致客户请求失败。

消息代理模式的不足之处在于：

- ✎ 消息驱动 Bean 没有返回值。这样通知客户执行结果只能依赖于 E-mail 或人工等其他手段。
- ✎ 消息驱动 Bean 执行过程中无法将捕获的异常直接返回给客户端，即无法使客户端直接知道错误信息。
- ✎ 消息驱动 Bean 通过接收 Message 来响应客户请求，对 Message 内容的合法性（比如对象的类型等）检验依赖于客户端，因此容易产生运行时错误。同时还要考虑由此带来的安全风险。

知识点索引

EJB；实体 EJB；会话 EJB；消息驱动 EJB；Servlet；

例程 6-2：基于 Struts 构建新闻发布系统

目的

掌握如何利用 Struts 来构建企业应用系统。

问题

如何构建一个新闻发布系统满足以下要求：

- (1) 功能需求：新闻发布用户的登录验证，新闻发布，新闻修改，新闻删除。
- (2) 非功能需求：系统具有良好的兼容性和可扩展性。

解决方案

利用 Struts 架构来构建新闻发布系统，通过 Struts 的基础架构将系统的表示逻辑层和业务逻辑进行有机配置，确保了系统的可扩展性；利用 Struts 的 Tile 子框架来组合前端页面，提高应用的可维护性。

知识链接

1. Web 应用开发模型

对于 Java 阵营的动态 Web 编程技术而言，应用体系架构经历了所谓的 Model 1 和 Model 2 时代。

(1) Model 1

在 Model 1 模式下，如图 6-7 所示，整个 Web 应用几乎全部由 JSP 页面组成，JSP 页面接收处理客户端请求，对请求处理后直接做出响应。用少量的 JavaBean 来处理数据库连接、数据库访问等操作。

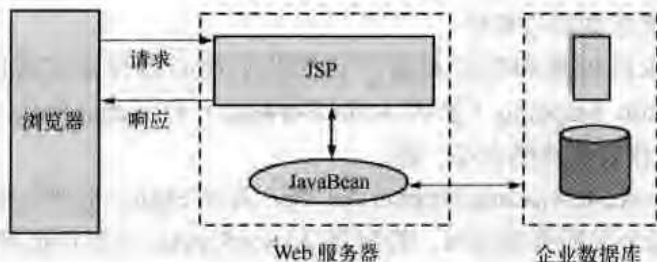


图 6-7 Model 1 模式

Model 1 使用标准的 Java EE 技术，最大的优点是实现简单，但它的缺点也很明显，主要表现为以下几方面。

- ✎ HTML 和 Java 混合降低开发效率：JSP 文件的编写者必须既是网页设计者，又是 Java 开发者。这对开发人员提出较高的要求，同时导致开发效率大大降低。
- ✎ 调试困难：除了很糟的外观之外，HTML 标记、Java 代码和 JavaScript 代码都集中在一个网页中，使调试变得相当困难。
- ✎ 表示逻辑与业务逻辑的强耦合：这也是 Model 1 最大的缺点，更改业务逻辑或数据可能牵涉相关的每个网页，导致程序变得笨重，难以维护。

(2) Model 2

Model 2 又被称为 MVC 模型。在 Model 2 应用体系中，从逻辑上将应用程序划分为以下 3 部分。

Model（模型）：包含应用程序的核心功能。它封装了应用程序的状态，有时它包含的唯一功能就是状态。模型对视图或控制器一无所知。

View（视图）：提供模型的表示，它是应用程序的外观。视图可以访问模型的读方法，但不能访问

写方法。此外，视图对控制器一无所知。当更改模型时，视图应得到通知。

Controller (控制器)：对用户的输入作出反应，它创建并设置模型。

在 Model 2 模式下，模型 (Model) 由 JavaBean 充当，视图 (View) 由 JSP 页面充当，而控制器 (Controller) 则由 Servlet 充当。在 Model 2 下，JSP 不再承担控制器的责任，它仅仅是表现层角色，用于将结果呈现给用户，JSP 页面的请求与 Servlet (控制器) 交互，而 Servlet 负责与后台的 JavaBean 通信。

由于引入了 MVC 模式，使 Model 2 具有组件化的特点，更适用于大规模应用的开发，但也增加了应用开发的复杂程度。原本需要一个简单的 JSP 页面就能实现的应用，在 Model 2 中被分解成多个协同工作的部分，需花更多时间才能真正掌握其设计和实现过程。那么如何设计规划系统的组件结构及它们之间的接口联系又摆在了软件架构师的面前。别着急，Struts 来了！

2. Struts 架构

Struts 始于 2000 年 3 月，是采用 Java Servlet/JavaServer Pages 技术开发 Web 应用程序的开放源码的框架。当前最新的正式版本是 2.0。采用 Struts 能迅速开发出基于 MVC 设计模式的 Java Web 前端应用。

Struts 由一组相互协作的类、Servlet 及 JSP TagLib 组成，分别用来实现模型、视图和控制器功能。基于 Struts 构架的 Web 应用程序完全符合 JSP Model 2 的设计标准，可以说是 MVC 设计模式的一种变化类型。

Struts 架构同时具有良好的扩展性，可以整合其他的一些技术去实现模型层 (Model) 和视图层 (View)。在模型层，Struts 可以很容易地与数据访问技术相结合，包括 EJB, JDBC 等。在视图层，Struts 能够与 JSP, Velocity Templates, XSL 等这些表示层组件结合。

3. Struts 框架的工作原理和核心组件

Struts 对于客户端请求的处理和响应是通过它的四个核心组件来实现的。这四个核心组件是：ActionServlet, Action, Action Mapping (包括 ActionForward) 和 ActionForm Bean。这四个组件通过配置文件 Struts-Config.xml 文件有机地结合在一起。

ActionServlet 继承自 javax.servlet.http.HttpServlet 类，其在 Struts 框架中扮演的角色是中心控制器。它提供一个中心位置来处理全部的终端请求。控制器 ActionServlet 主要负责将 HTTP 的客户请求信息组装后，根据配置文件的指定描述，转发到适当的处理器。

Action 类担负模型的角色，就像客户请求动作和业务逻辑处理之间的一个适配器 (Adaptor)，其功能就是将请求与业务逻辑分开。

Action Mapping 负责应用程序控制流程的转向，其中 ActionForward 对象封装了向前进的 URL 路径且被 ActionServlet 用于识别目标视图。

ActionForm Bean 负责保持一个应用系统的消息转移 (或者说状态转移) 的非持久性数据存储。

用户的请求是通过 ActionServlet 来处理 and 转发的。那么，ActionServlet 如何决定把用户请求转发给哪个 Action 对象呢？这时需要一些描述用户请求路径和 Action 映射关系的配置信息。在 Struts 中，这些配置映射信息都存储在特定的 XML 文件——Struts-config.xml 中。这些配置信息在系统启动时被读入内存，供 Struts 在运行期间使用。

对于采用 Struts 框架的 Web 应用，在 Web 应用启动时就会自动加载并初始化 ActionServlet，ActionServlet 从 struts-config.xml 文件中读取配置信息，在收到客户请求时将根据这些配置信息来决定程序的下一步动作。具体流程如下：

(1) 检索与用户请求相匹配的 ActionMapping 实例，如果不存在，就返回用户请求路径无效信息。

(2) 如 ActionForm 实例不存在，就创建一个 ActionForm 对象，把客户提交的表单数据保存到 ActionForm 对象中。

(3) 根据配置信息决定是否需要表单验证。如果需要验证,就调用 ActionForm 的 Validate()方法。

(4) 如果 ActionForm 的 Validate()方法返回 null 或返回一个不包含 ActionMessage 的 ActionErrors 对象,就表示表单验证成功。

(5) ActionServlet 根据 ActionMapping 实例包含的映射信息决定将请求转发给哪个 Action。如果相应的 Action 实例不存在,就先创建这个实例,然后调用 Action 的 execute()方法。

(6) Action 的 execute()方法返回一个 ActionForward 对象,ActionServlet 再把客户请求转发给 ActionForward 对象指向的 Web 组件。

(7) ActionForward 对象指向的组件生成动态网页,返回给客户。

运行准备

1. 创建后台数据库

打开 MySQL 数据库,建立一个新闻发布系统专用的数据库 news。系统中仅有 3 个表格:表 news_info 用来保存新闻信息,表 channel 用来保存新闻频道信息,表 user 用来保存用户信息。表的详细信息分别如表 6-1、表 6-2 和表 6-3 所示。

表 6-1 用户信息表 user 的结构信息

字段名	字段类型	备注	字段说明
Id	int(6)	主键,系统自增字段	用户 ID
name	varchar (10)		用户登录名称
password	varchar (10)		用户密码

表 6-2 频道信息表 channel 的结构信息

字段名	字段类型	备注	字段说明
channel_id	int(6)	主键,系统自增字段	频道 ID
channel_name	varchar (50)		频道名称
channel_order	int(1)		频道顺序
channel_status	int(1)		频道状态

表 6-3 新闻信息表 news_info 的结构信息

字段名	字段类型	备注	字段说明
new_id	int(10)	主键,系统自增字段	新闻 ID
channle_id	int(10)		频道 ID
subject	varchar (100)		标题
create_time	datetime		创建时间
content	longtext		内容
author	varchar(50)		作者

提示:可以根据上述信息创建相应的数据库表格,也可以从脚本文件 news.sql 中导入数据库信息。

2. 为应用程序添加 Struts 支持

Netbeans 为开发基于 Struts 的 Web 应用提供了强大支持,可以在创建项目的过程中直接创建基于 Struts 应用。单击【文件】→【新建项目】,弹出【新建项目】对话框,如图 6-8 所示。



图 6-8 【新建项目】对话框

在【类别】列表中选择“Web”，在项目列表中选择【Web 应用程序】来创建一个 Web 应用。单击【下一步】按钮进入下一步，得到如图 6-9 所示的【新建 Web 应用程序】对话框。



图 6-9 【新建 Web 应用】对话框

在【项目名称】文本框中输入项目的名称“struts_new”，默认其他选项设置，单击【下一步】按钮，得到如图 6-10 所示的对话框。选择【Struts 1.2.9】及【添加 StrutsTLD (A)】复选框，其他选项按照默认设置。最后单击【完成】按钮，一个 Struts 应用程序的创建完成。



图 6-10 为项目添加 Struts 支持

Netbeans 为 Web 应用添加的 Struts 支持主要表现在以下两个方面。

(1) 为 Web 项目添加了所需要的库。在【项目】视图中单击【库】文件夹，可以看到 Struts 所需的库已经添加进来，如图 6-11 所示。

(2) 为 Web 项目添加了所需要的配置文件。在【项目】视图中单击【配置】文件夹，可以看到自动生成的配置文件。如图 6-12 所示。除了 Web 应用程序标准的配置文件外，还多了 struts-config.xml、tiles-def.xml、validation.xml 和 validator-rules.xml 文件。

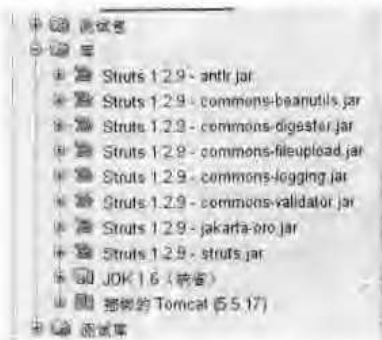


图 6-11 Struts 框架所需的库



图 6-12 Struts 配置文件

其中，struts-config.xml 是整个 Struts 架构的配置文件，validation.xml 和 validator-rules.xml 用来对系统进行校验，tiles-def.xml 定义页面组合规则。关于上述配置文件的具体配置方法在后面还会详细论述。

双击配置文件 web.xml，可以看到 Struts 控制器 ActionServlet 和标记库的配置信息已经自动添加到配置文件中（如斜体所示）。

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <init-param>
      <param-name>debug</param-name>
      <param-value>2</param-value>
    </init-param>
    <init-param>
      <param-name>detail</param-name>
      <param-value>2</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
```



```

</servlet-mapping>
<session-config>
    <session-timeout>
        30
    </session-timeout>
</session-config>
<welcome-file-list>
    <welcome-file>
        index.jsp
    </welcome-file>
</welcome-file-list>
<jsp-config>
    <taglib>
        <taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
        <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
    </taglib>
    <taglib>
        <taglib-uri>/WEB-INF/struts-html.tld</taglib-uri>
        <taglib-location>/WEB-INF/struts-html.tld</taglib-location>
    </taglib>
    <taglib>
        <taglib-uri>/WEB-INF/struts-logic.tld</taglib-uri>
        <taglib-location>/WEB-INF/struts-logic.tld</taglib-location>
    </taglib>
    <taglib>
        <taglib-uri>/WEB-INF/struts-nested.tld</taglib-uri>
        <taglib-location>/WEB-INF/struts-nested.tld</taglib-location>
    </taglib>
    <taglib>
        <taglib-uri>/WEB-INF/struts-tiles.tld</taglib-uri>
        <taglib-location>/WEB-INF/struts-tiles.tld</taglib-location>
    </taglib>
</jsp-config>
</web-app>

```

实现步骤

1. 配置 Struts 数据源

为了提高程序的移植性及数据库的访问效率，可通过配置数据源来访问数据库。具体操作如下。

(1) 将与数据源访问相关的库添加到工程的类路径下。具体操作为：在【项目】视图下，选中项目文件夹【Struts_new】下的【库】文件夹，单击右键，在弹出的快捷菜单中选择【添加 JAR/文件夹】选项。需要添加的库包括：commons-dbcp-1.2.1.jar，commons-pool-1.2.jar，commons-collection.jar 和 struts-legacy.jar。另外，为了访问数据库，还需要将 MySQL 的 JDBC 驱动程序添加到工程的类路径下。

(2) 在 struts-config.xml 文件下添加数据源配置信息。在本例中要添加的配置信息如下：

```

<data-sources>
    <data-source key="MySQL_datasource" type="org.apache.commons.dbcp.BasicDataSource">
        <set-property property="driverClassName" value="org.gjt.mm.mysql.Driver" />
        <set-property property="minCount" value="2" />
    </data-source>
</data-sources>

```

```

<set-property property="maxCount" value="10" />
<set-property property="description" value="datasource sample" />
<set-property property="username" value="root" />
<set-property property="password" value="javaee" />
<set-property property="url" value="jdbc:mysql://localhost/struts_news?
autoReconnect=true&useUnicode=true&characterEncoding=utf-8" />
<set-property property="autoCommit" value="false" />
<set-property property="readOnly" value="false" />
</data-source>
</data-sources>

```

其中在属性 url 的值中,还指定了数据库连接时的编码格式为 utf-8。

2. 利用 Struts 实现用户登录模块 (Struts 开发基本流程)

用户登录模块比较简单,主要包括一个登录界面,一个代表用户提交信息的 JavaBean 对象,一个后台登录验证组件,还有一个登录结果的显示页面。我们将通过 Struts 框架将上述组件集成在一起,演示如何利用 Struts 框架来架构应用。

(1) 创建登录界面 login.jsp。完整的源代码如程序 6-6 所示。

(2) 创建 FormBean LoginForm 来代表用户提交的信息。完整的源代码如程序 6-7 所示。

(3) 在配置文件 struts-config.xml 中添加 LoginForm 的配置信息。配置信息如程序 6-8 所示。

(4) 创建数据操作接口 loginDAO.java 来抽象登录模块对数据库的操作。完整的源代码如程序 6-9 所示。

(5) 创建数据库操作 loginDAOImpl.java 来实现登录模块对数据库的操作。完整的源代码如程序 6-10 所示。

(6) 创建 Action 组件 LoginAction 来调用数据库操作组件实现登录处理。完整的源代码如程序 6-11 所示。

(7) 在配置文件 struts-config.xml 中添加 LoginAction 配置信息。配置信息如程序 6-12 所示。

(8) 创建登录成功提示页面 success.jsp,如程序 6-13 所示。

3. 应用 tiles 子框架定义复杂页面组合

在许多 Web 应用中,界面往往是由一组页面组成。在 Struts 中,提供了 tiles 子框架来帮助开发人员定义复杂页面组合。在本例中,工作主界面由导航栏、显示主体和版权信息三部分组成。下面通过 tiles 子框架来实现它。具体步骤如下。

(1) 在 struts-config.xml 文件中配置 Tiles 插件,具体代码如下所示:

```

<plug-in className="org.apache.struts.tiles.TilesPlugin" >
    <set-property property="definitions-config" value="/WEB-INF/tiles-defs.xml" />
    <set-property property="moduleAware" value="true" />
</plug-in>

```

其中属性 definitions-config 定义了 tiles 插件使用的配置信息存放在文件 tiles-defs.xml 中。

(2) 创建模板页面文件 template_main.jsp。为了对组合的页面进行精确控制,仍然使用了框架。完整源代码如程序 6-7 所示。

(3) 基于模板页面文件创建 tiles 组合页面,如代表应用工作主界面的 front.jsp,代码如程序所示。

4. 实现新闻展示与管理模块

主要包括新闻的展示,新闻的发布等功能。为节省篇幅,就不一一赘述。详细内容请参考随书附带的源代码。

示例代码

本例程的所有代码都包含在工程 `struts_new` 中。

程序 6-6: `login.jsp`

```
<%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
<%
String path = request.getContextPath();
String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.
getServerPort()+path+"/";
%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<base href="<%=basePath%>">
<title>欢迎登录千里眼新闻系统</title>

<meta http-equiv="pragma" content="no-cache">
<meta http-equiv="cache-control" content="no-cache">
<meta http-equiv="expires" content="0">
<meta http-equiv="keywords" content="keyword1,keyword2,keyword3">
<meta http-equiv="description" content="This is my page">

<link href="./css/style.css" rel="stylesheet" type="text/css">
</head>

<body>
<form action="login.do" method="post">

<TABLE width="100%">
<TR>
<TD align="center">用户名: <input type="text" name="userName"></TD>
</TR>
<TR>
<TD align="center">密 码: <input type="password" name="passWord"></TD>
</TR>
<TR>
<TD align="center"><input type="submit" value="登录"><input type="reset" value="
重置"></TD>
</TR>
</TABLE>
</form>
</body>
</html>
```

程序说明: 页面主要用来搜集用户登录信息。

程序 6-7: `LoginForm.java`

```
package com.news.struts.form;
```

```

import javax.servlet.http.HttpServletRequest;
import org.apache.struts.action.ActionErrors;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionMapping;
public class LoginForm extends ActionForm {
    private String userName;
    private String passWord;
    public ActionErrors validate(
        ActionMapping mapping,
        HttpServletRequest request) {

        // TODO Auto-generated method stub
        return null;
    }

    public void reset(ActionMapping mapping, HttpServletRequest request) {

        // TODO Auto-generated method stub
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPassWord() {
        return passWord;
    }

    public void setPassWord(String passWord) {
        this.passWord = passWord;
    }
}

```

程序说明：用来封装用户提交的数据信息，并提供了对这些信息的操作方法。由于用户提交的登录信息只有两个：userName 和 passWord。

程序 6-8: Struts-config.xml (LoginForm 的配置信息)

```

...
<form-beans>

    <form-bean name="LoginForm" type="com.news.struts.form.LoginForm" />
</form-beans>
...

```

程序 6-9: LoginDAO.java


```

package com.news.DAO;

import com.news.struts.form.LoginForm;
import java.sql.Connection;

public interface LoginDAO {
    public boolean check (LoginForm loginform) throws Exception;
    public void setConnection(Connection conn);
    public Connection getConnection();
}

```

程序说明：定义登录模块对数据库的操作接口。

程序 6-10: LoginDAOImpl.java

```

package com.news.DAO.impl;

import com.news.entity.News;
import com.news.struts.form.LoginForm;
import com.news.util.DTOPopulator;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.List;

public class LoginDAOImpl {
    Connection connection = null;
    public LoginDAOImpl() {
    }
    public Connection getConnection() {
        return connection;
    }
    public void setConnection(Connection connection) {
        this.connection = connection;
    }
    public boolean check (LoginForm loginform) throws Exception{
        String name=loginform.getUserName();
        String password=loginform.getPassWord();
        PreparedStatement ps = connection.prepareStatement("select * from user
        where name= ? and password = ? ");

        ps.setString(1,name);
        ps.setString(2,password);
        ResultSet rs = ps.executeQuery();
        if(rs.next()) return true;
        else return false;
    }
}

```

程序说明：具体实现用户登录时对数据库的操作。其中的属性 `connection` 用来代表从数据源获取的连接对象。

程序 6-11: LoginAction.java

```

package com.news.struts.action;

```

```

import com.news.DAO.LoginDAO;
import com.news.DAO.impl.LoginDAOImpl;
import com.news.struts.form.LoginForm;
import java.sql.SQLException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;
import org.apache.struts.action.Action;
import org.apache.struts.action.ActionForm;
import org.apache.struts.action.ActionForward;
import org.apache.struts.action.ActionMapping;
public class LoginAction extends Action{
    DataSource dataSource=null;
    public LoginAction() {
    }
    public ActionForward execute(
        ActionMapping mapping,
        ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response) {

        dataSource=getDataSource(request,"MySQL_Datasource");

        LoginForm lf=(LoginForm) form;
        LoginDAO dao=new LoginDAOImpl();
        try{
            dao.setConnection(dataSource.getConnection());
            if(dao.check(lf))return mapping.findForward("success");
            else return mapping.findForward("fail");
        }catch(SQLException e){
            System.out.println(e.toString());
        }catch(Exception e){
            System.out.println(e.toString());
        }
        return mapping.findForward("fail");
    }
}

```

程序说明：程序的主要业务逻辑都在 `execute()` 方法中实现。由于已经配置好了数据源，那么在代码中可以直接调用 `getDataSource()` 方法来获取数据源对象。根据获取的数据源对象获得数据库的连接并传递到 `LoginDAOImpl` 中，然后根据方法的执行结果来确定不同的业务流向。

程序 6-12: `template_main.jsp`

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<%@ page language="java" contentType="text/html; charset=GB2312"%>
<%@ taglib uri="http://jakarta.apache.org/struts/tags-bean" prefix="bean" %>
<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles"%>
<tiles:importAttribute/>
    <bean:define id="Header" name="header" type="String"/>
    <bean:define id="Copyright" name="copyright" type="String"/>

```



```

<bean:define id="BodyContent" name="body" type="String"/>

<html>
<head>
<title>新闻</title>
<meta http-equiv="Content-Type" content="text/html; charset=GB18030">
</head>

<frameset rows="40,*" cols="*" frameborder="NO" border="0" framespacing="0">
  <frame name="topFrame" scrolling="NO" noresize src="<%=Header%>" >
  <frameset rows="90%,10%" cols="*" frameborder="NO" border="0" framespacing="0">
    <frame name="main" scrolling="auto" src="<%=BodyContent%>">
    <frame name="copyright" scrolling="NO" src="<%=Copyright%>">
  </frameset>
</frameset>

<noframes><body bgcolor="#FFFFFF" text="#000000">

</body></noframes>
</html>

```

程序说明：作为 tiles 框架的模板文件，必须通过语句 `<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles"%>` 引入 tiles 标记库，并通过 `<tiles:importAttribute/>` 标记插入 tiles 属性标记。

程序 6-13: front.jsp

```

<%@ taglib uri="/WEB-INF/struts-tiles.tld" prefix="tiles" %>
<tiles:insert page="/layout/template_main.jsp" flush="true">
  <tiles:put name="header" value="Channle.do?op=list"></tiles:put>
  <tiles:put name="body" value="main.jsp"></tiles:put>
  <tiles:put name="copyright" value="common/Copyright.html"></tiles:put>
</tiles:insert>

```

程序说明：通过 tiles 框架创建复合页面。首先引入 tiles 标记库，然后利用 insert 标记和 put 标记将实际页面的 URL 地址插入到模板文件中。

运行结果

程序发布成功后，在浏览器地址栏中输入“`http://localhost:8084/Struts_new/`”，将得到如图 6-13 所示的运行结果。



图 6-13 用户登录界面

输入用户名和密码, (在本实例中用户名和密码分别为 test 和 test), 将得到如图 6-14 所示的运行界面。



图 6-14 登录成功后显示的首页

单击想要查看的频道栏目, 则可以看到相应的新闻列表, 如图 6-15 所示。



图 6-15 新闻列表页面

讨论与思考

Struts 最大的优点在于下面两点。

(1) 通过标准的组件实现了 Web 应用开发中的控制信息流和数据信息流的标准化处理, 规范了开发人员的行为, 提高了系统的可维护性。其中对控制流的处理主要通过 ActionServlet、Actionmapping 来实现, 对数据流的处理主要通过 ActionForm 来实现。各组件之间通过一个配置文件来结合。通过配置文件使系统的脉络更加清晰。开发人员可以很容易地把握整个系统各部分之间的联系, 这对于后期的维护有着莫大的好处。尤其是当另一批开发者接手这个项目时, 这种优势体现得更加明显。

(2) Struts 框架为开放者提供了一个 Taglib, 以及 Validator 子框架、Tiles 子框架等有用的功能特性, 通过灵活应用, 能大大提高开发效率。

Strats 的缺点在于, 层面太多, 对于较小的应用开发, 远不如 JSP+Bean 的方式简洁快速。

知识点索引

Struts; 数据源; Tiles; 架构。

例程 6-3：基于 Struts、Spring 和 Hibernate 构建学生信息管理系统

目的

演示如何通过集成 Struts、Spring 和 Hibernate 构建企业信息系统。

问题

如何构建一个学生信息管理系统，并满足以下要求：

- (1) 实现对学生基本信息的增加、删除和修改等操作；
- (2) 系统具有良好的扩展性和可维护性。

解决方案

为了尽可能提高系统各功能模块的独立性、扩展性，系统采用传统的三层设计结构，即：表示层、业务逻辑层和持久层，从而使得各个模块层上的变化尽可能少地影响其他模块，提高系统的扩展性和可维护性。Struts、Spring、Hibernate 框架分别在这三个层次上很好地满足了开发要求，并且它们都有完整的文档，相对来讲比较简单，所以用它们开发系统比较方便。

(1) Struts 应用在 UI 部分通过 JSP 技术实现。Struts 提供了自定义的标记库可以使用，通过这些自定义标记可以非常好地和系统交互，通过使用这些自定义标记创建的 JSP 表单，可以实现对用户数据的封装，同时这些自定义标记还提供了像模板定制等多种显示功能。因此用来开发 UI 较为方便。

(2) 对于一个管理系统，最主要的应该是后端数据的持久化，它关系到管理系统运行的可靠性和可扩展性。为了减少数据库变化对“业务层”的影响，在此选用 ORM 开源框架 Hibernate，它是一个开放源代码的对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，使得 Java 程序员可以随心所欲地使用对象编程思维来操纵数据库。Hibernate 允许将数据库中的信息存放入业务对象 (Domain Objects)，这样可以在连接断开的情况下把这些数据显示到 UI 层。而那些对象也可以返回给持久层，从而在数据库里更新。它为 Java 开发者提供了很好的对象关系持久化机制和查询服务。

(3) 一个典型 Web 应用的中间部分是业务层或者服务层。从编码的角度来看，这层是最容易被忽视的一层。往往在 UI 层或持久层嵌入这些业务处理的代码，导致了程序代码的紧密耦合，这样一来，随着时间推移这些代码就很难维护。因此针对这一问题采用了 Spring 框架，它能很好地把对象搭配起来。Spring 在设计时就充分考虑到与 Hibernate 的协同工作，通过内置的 Hibernate 支持在两者之间提供了良好的结合点。其优势主要体现在：

- ✎ 利用延时注入思想组装代码，提高系统扩展性，灵活性，实现插件式编程；
- ✎ 利用 (Aspect-Oriented Programming、面向方面编程) 思想，集中处理业务逻辑，减少重复代码，构建优雅的解决方案；
- ✎ 利用其对 Hibernate 的 SessionFactory、事务管理的封装，更简捷地应用 Hibernate。

知识链接

1. Hibernate 框架

为了企业应用与后端数据库频繁交互，并且使得交互更加有效而迅捷，企业应用开发者在应用和数据库之间创建了一个“持久层”。持久层负责存储从应用到数据库的数据，也负责数据的检索、更新和删除。在基于 Java EE 的企业应用中，组成这个持久层的 Java 类既可以映射对象到数据，也可以映射数据到对象。持久层的建立是简单的，但是，这种关系常常难于建立，因为对象或者下层的数据库结构复

杂, 很难做到把关系表记录完整地映射到持久对象的关系上来, 这主要体现在多表的关系无法直接映射到对持久对象的映射上来, 可能是一个表映射多个持久对象, 有可能是多个表映射一个持久对象, 更有可能是表的某些字段映射到一个持久对象, 另外一些字段映射到别的持久对象上。

Hibernate 是一个开放源代码的 O/R Mapping (对象关系映射框架), 它对 JDBC 进行了轻量级的对象封装, 使 Java 程序员可以随心所欲地使用对象编程思维来操纵数据库。Hibernate 帮助基于普通的 Java 对象模型的持久对象的创建, 从而允许持久对象拥有复杂的结构, 如混合类型、集合和属性, 还可以拥有用户自定义的类型。现在这些持久对象可以有效地反映出底层数据库模式的复杂结构。为了提高效率, Hibernate 包括了一些策略, 如与数据库交互时的多重最优化, 包括对象的缓存、有效外部连接的获取、必要时 SQL 语句的执行。

Hibernate 配置文件可以有两种格式, 一种是 hibernate.properties, 另一种是 hibernate.cfg.xml。后者稍微方便一些, 当增加.hbm 映射文件的时候, 可以直接在 hibernate.cfg.xml 里面增加, 不必像 hibernate.properties 必须在初始化代码中加入。

2. Spring 框架

Spring 作为实现 Java EE 的一个全方位应用程序框架, 为开发企业级应用提供了一个健壮、高效的解决方案。它主要有以下几个特点。

(1) 非侵入式

所谓非侵入式是指 Spring 框架的 API 不会在业务逻辑上出现, 也就是说业务逻辑应该是纯净的, 不能出现与业务逻辑无关的代码。首先针对应用而言, 这样才能将业务逻辑从当前应用中剥离出来, 从而在其他的应用中实现复用; 其次针对框架而言, 由于业务逻辑中没有 Spring 的 API, 所以业务逻辑也可以从 Spring 框架快速移植到其他框架。

(2) 容器

Spring 提供容器功能, 容器可以管理对象的生命周期, 对象与对象之间的依赖关系。用户可以写一个配置文件 (通常是 XML 文件), 在上面定义对象的名字, 是否为单例, 以及设置与其他对象的依赖关系。那么在容器启动之后, 这些对象就被实例化了, 用户可直接使用, 而且依赖关系也建立好了。

(3) IOC

控制反转, 谓之“依赖关系的转移”, 如果以前都是依赖于实现, 那么现在反转为依赖于抽象, 其实它的核心思想就是要面向接口编程。

(4) 依赖注入

建立对象与对象之间依赖关系的实现, 包括接口注入、构造注入、set 注入, 在 Spring 中只支持后两种。

(5) AOP

面向方面编程, 我们可以把日志、安全、事务管理等服务 (或功能) 理解成一个“方面”, 那么以前这些服务一直是直接写在业务逻辑的代码当中的, 这有两点不好: 首先业务逻辑不纯净, 其次这些服务被很多业务逻辑反复使用, 完全可以剥离出来做到复用。那么 AOP 就是这些问题的解决方案, 我们可以把这些服务剥离出来形成一个“方面”, 以期做到复用; 然后将“方面”动态地插入到业务逻辑中让业务逻辑能够享受到此“方面”的服务。

Spring 还有其他一些功能: 对 JDBC 的封装与简化, 提供事务管理功能, 对 O/R Mapping 工具 (Hibernate、iBATIS) 的整合; 提供 MVC 解决方案, 也可以与其他 Web 框架 (Struts、JSF) 进行整合; 还有对 JNDI、Mail 等服务进行封装等。

运行准备

(1) 下载 Spring 地址为 <http://www.springframework.org/download>。本书使用的版本是 1.2.6。

(2) 下载 Hibernate: 地址为 <http://www.hibernate.org>。本书使用的版本是 3.0。

(3) 下载 Struts: Netbeans 开发环境已经内置了对 Struts 的支持, 当然也可以去地址 <http://struts.apache.org> 下载新的版本。本书使用的版本是 1.2。

(4) 为了使用 Apache 的 DBCP 数据源, 还要下载 common-pool.jar, common-beanutil.jar 和 struts-legacy.jar。

(5) 为了访问 MySQL 数据库, 还要下载 MySQL 数据库驱动程序。

实现步骤

1. 创建基于 Struts 的 Web 应用

单击【文件】→【新建项目】, 弹出【新建项目】对话框。在【类别】中选择“Web”然后在【项目】中选择“Web 应用程序”创建一个 Web 应用。单击【下一步】, 打开【新建 Web 应用程序】对话框。在【项目名称】文本框中输入为“Manager”, 在【服务器】下拉列表中选择“Sun Java System Application Server 9”, 如图 6-16 所示。



图 6-16 创建基于 Struts 的 Web 应用

单击按钮【下一步】, 选择“Struts 1.2.9”及“添加 Struts TLD”, 如图 6-17 所示, 单击【完成】按钮, 则支持 Struts 的 Web 应用程序的创建完毕。

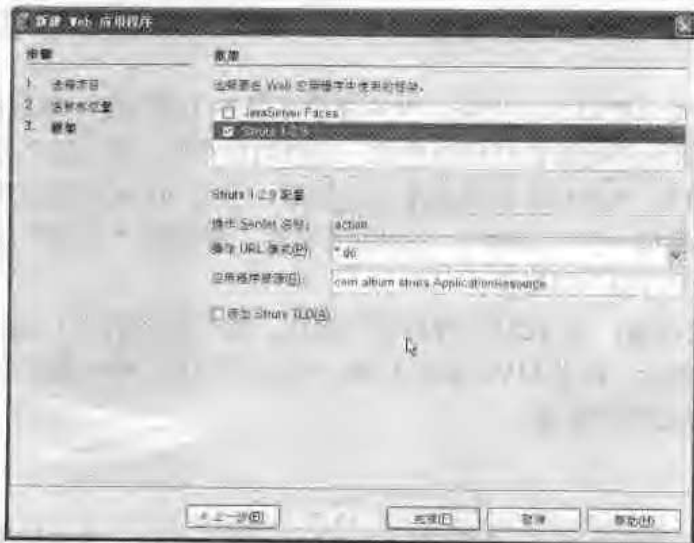


图 6-17 配置 Struts 信息

2. 添加 Hibernate 支持

由于 Spring 内置了对 Hibernate 的支持, 关于 Hibernate 的数据源配置等信息都集成到 Spring 的配置文件中, 因此只需要解压下载的 Hibernate 包, 并将 Hibernate3.jar 添加到项目的类路径下即可。

生成数据库相关表格的 Hibernate 映射文件。关于 Hibernate 对数据库的映射文件的生成, 请参考相关的资料。

3. 添加 Spring 支持

解压下载的 Spring 包, 并将 Spring 相关的 jar 文件添加到项目的类路径下。具体需要哪些 jar 可查看详细源代码。

为了使 Web 应用程序支持 Spring 框架, 还必须修改配置文件 Web.xml, 使得 Web 应用在初始化是自动启动 Spring 框架。具体配置信息如下:

```
...
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/hibernate_context.xml</param-value>
</context-param>
<listener>
    <listener-class>com.student.listener.StartupListener</listener-class>
</listener>
...
```

同时在路径/web-inf/下添加一个 Spring 框架的初始化配置文件 hibernate_context.xml。

说明: Spring 初始化方式有两种。一种是基于监听器的初始化方法, 即通过配置监听器 org.springframework.web.context.ContextLoaderListener 确保在 Web 应用启动时初始化 Spring 框架, 本示例即采用此方法, com.student.listener.StartupListener 继承了 org.springframework.web.context.ContextLoaderListener。另外一种是基于 Servlet 的初始化方法, 即通过配置 ContextLoaderServlet 来确保在 Web 应用启动时初始化 Spring 框架, 示例配置信息如下所示:

```
<servlet>
<servlet-name>Context</servlet-name>
<servlet-class>org.springframework.web.context.ContextLoaderServlet</servlet-
class>
<load-on-startup>1</load-on-startup>
</servlet>
```

其中<load-on-startup>使得 Servlet 在 Web 应用启动时被容器自动加载。

上述两种方式没有本质的区别。

完整的 Web 应用配置信息如程序 6-16 所示。

4. Spring 与 Hibernate 的整合

Spring 对 Hibernate 的集成提供了定义 Hibernate 资源和对 Hibernate 资源的依赖注入。

(1) 定义 Hibernate 资源

它是通过 Spring 框架的配置文件中 hibernate_context.xml 来获取 Hibernate 会话资源。具体操作如下: 首先配置 dbcp 数据源, 然后利用此数据源作为属性参数来创建 Bean sessionFactory, 它是 org.springframework.orm.hibernate3.LocalSessionFactoryBean 的一个实例, 主要用来获取 Hibernate 的会话对象。sessionFactory 的另一个参数是代表各种 Hibernate 数据实体对象的映射文件列表。

(2) 对 Hibernate 资源的依赖注入

在 Spring 的配置文件 hibernate_context.xml 中。任何需要 Hibernate 资源的地方，只需要引用 Bean sessionFactory 即可。如：

```
<bean id="studentDAOHibernate" class="com.student.dao.hibernate.StudentDAOHibernate">
    <property name="sessionFactory"><ref local="sessionFactory"/></property>
</bean>
```

这样，就可以使用 Spring 的 IOC 特性来管理各 Bean。在这种模式下，Spring 的 IOC 模式可以统一管理各层，而又使各层松散耦合在一起，使各层之间实现最大的解耦性，这也是 Web 架构一向的追求。

5. 利用 Spring AOP 实现事务支持

软件系统通常由多个组件构成，每个组件负责一个特定的功能领域。但是，这些组件也经常承担它们的核心功能之外的额外责任，如业务逻辑组件在实现业务逻辑的同时还要确保当发生意外时进行事务回滚操作等，结果就造成不同领域的代码紧密耦合在一起。面向方面编程是一种试图解决这个问题的编程技术。

所谓的“方面”，是一些模块化的通用的功能特性，如日志和事务管理等。显然，可以把这些服务的支持直接编写到要求服务的每个类当中，但是更希望能够不必为大量事务性上下文编写同样的事务处理代码。

基于 Spring AOP 时，仍然是像创建普通的 Bean 一样在配置文件中的某处定义系统的公共功能，但是可以声明性地定义“如何”和“在哪里”应用这个功能。那么对于其他 Bean 的创建，Spring 框架会根据配置信息自动向代码中添加新特性，从而避免了烦琐的编码。这便是面向方面的编程思想。

本实例中以事务支持为例演示了 Spring AOP 特性。

在配置文件中定义 Bean 如下：

```
<bean abstract="true"
    class="org.springframework.transaction.interceptor.
    TransactionProxyFactoryBean"
    id="baseTransactionProxy">
    <property name="transactionManager">
        <ref bean="transactionManager" />
    </property>
    <property name="transactionAttributes">
        <props>
            <prop key="save*">PROPAGATION_REQUIRED</prop>
            <prop key="update*">PROPAGATION_REQUIRED</prop>
            <prop key="find*">PROPAGATION_REQUIRED</prop>
            <prop key="remove*">PROPAGATION_REQUIRED</prop>
        </props>
    </property>
</bean>
```

通过上面的配置信息，抽象类 baseTransactionProxy 继承了 TransactionProxyFactoryBean，能够拦截对 transactionManager 的方法调用，并把事务上下文应用到其中。

通过随后定义代表业务层服务的 Bean manager 并继承抽象类 baseTransactionProxy（代码如程序 6-13 所示），将事务支持特性注入到业务层实现代码中，这样在 manager 的实现中就不需要编写任何事务支持相关的代码即可支持事务功能特性，这样就实现了基于“方面”开发。

6. Spring 与 Struts 的集成

通过 `org.springframework.web.struts.DelegatingRequestProcessor` 类充当 Action 的代理, 将 Struts Action 配置在 Spring `ApplicationContext` 中, 每次客户请求 Struts Action 时, `DelegatingRequestProcessor` 将充当代理的作用, 即通过它将 Action 请求转发给 Spring IOC 容器进行处理。Spring IOC 容器根据配置信息将生成相应的 Action 实例来处理客户端的请求。

如此一来, Struts 在运行期间加载的实际上是 `DelegatingActionProxy`, 而 `DelegatingActionProxy` 则实现了针对实际 Action 的调用代理, Struts 最终调用的将是由 Spring 管理的 Action 实例。通过这样的方式, Spring 获得了对 Action 实例的管理权, 它将对 Action 进行调度, 并为 Struts 提供所需的 Action 实例。既然 Action 已经由 Spring 全权接管, 那么就可以将此 Action 看作是 Spring 中的一个 Bean, 它可享受 Spring 提供的所有服务 (依赖注入、实例管理、事务管理等)。

具体步骤如下:

- (1) 在 Struts-config 文件中配置 Spring 请求代理插件, 配置信息如下:

```
...
<message-resources parameter="com.student.application"/>
  <controller processorClass="org.springframework.web.struts.
    DelegatingRequestProcessor"/>
  <plug-in className=
    "org.springframework.web.struts.ContextLoaderPlugIn">
    <set-property property=
      "contextConfigLocation" value="/WEB-INF/hibernate_context.
        xml"/>
  </plug-in>
...
```

- (2) 在 Spring 配置文件中注册 Action。在本例中的配置信息如下:

```
...
<bean name="/studentAction"
  class="com.student.webapp.action.StudentManageAction">
  <property name="studentManager"> <ref bean="studentManager"/>
  </property>
</bean>
...
```

通过将 Action 注册为 Spring IOC 容器管理的 Bean, 在客户端发成请求 `/studentAction` 时, 通过代理类 `DelegatingActionProxy` 的拦截, 请求交由 Spring 框架处理, Spring IOC 容器则负责创建并管理对应的 Action 的实例, 由此实例处理客户端请求。因此, 此时注册的 Bean 的名称必须与客户端请求的 URL 相匹配。

说明: 由于属性 id 不允许以 “/” 出现, 因此, 这里使用 name 属性。

(3) 生成具有 JavaBean 特性的 Action。为了支持 Spring IOC 容器的依赖注入, 必须首先将 Action 中用到的其他实例定义为变量, 并提供相应的 `getter/setter` 方法。这样就不必在 Action 中显式地创建变量, 而是由 Spring IOC 容器根据配置信息生成注入, 降低程序的耦合度。本例中的 `StudentManagerAction` 的源代码如程序 6-14 所示。

示例代码

本例程的所有代码都包含在工程 Manager 中。

程序 6-14: hibernate_context.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN" "http://www.springframework.
org/dtd/spring-beans.dtd">
<beans>

    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">

        <property name="driverClassName">
            <value>com.mysql.jdbc.Driver</value>
        </property>

        <property name="url">
            <value>jdbc:mysql://localhost/student</value>
        </property>

        <property name="username">
            <value>root</value>
        </property>

        <property name="password">
            <value>javaee</value>
        </property>
    </bean>

    <bean id="sessionFactory" class="org.springframework.orm.hibernate3.
LocalSessionFactoryBean">

        <property name="dataSource">
            <ref local="dataSource" />
        </property>

        <property name="mappingResources">
            <list>
                <value>com/student/model/Student.hbm.xml</value>
            </list>
        </property>

        <property name="hibernateProperties">
            <props>
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQLDialect
                </prop>
                <prop key="hibernate.show_sql">true</prop>
                <prop key="hibernate.max_fetch_depth">1</prop>
            </props>
        </property>
    </bean>

```

```

<bean id="transactionManager" class="org.springframework.orm.hibernate3.
HibernateTransactionManager">
    <property name="sessionFactory">
        <ref local="sessionFactory" />
    </property>
</bean>

<bean id="jdbcExceptionTranslator" class="org.springframework.jdbc.support.
SQLExceptionCodeSQLExceptionTranslator">
    <property name="dataSource">
        <ref bean="dataSource" />
    </property>
</bean>

<!-- Hibernate Template -->
<bean id="hibernateTemplate" class="org.springframework.orm.hibernate3.
HibernateTemplate">
    <property name="sessionFactory">
        <ref bean="sessionFactory" />
    </property>
    <property name="jdbcExceptionTranslator">
        <ref bean="jdbcExceptionTranslator" />
    </property>
</bean>

<!--baseTransactionProxy -->
<bean abstract="true" class="org.springframework.transaction.interceptor.
TransactionProxyFactoryBean"
id="baseTransactionProxy">
    <property name="transactionManager">
        <ref bean="transactionManager" />
    </property>
    <property name="transactionAttributes">
        <props>
            <prop key="save*">PROPAGATION_REQUIRED</prop>
            <prop key="update*">PROPAGATION_REQUIRED</prop>
            <prop key="find*">PROPAGATION_REQUIRED</prop>
            <prop key="remove*">PROPAGATION_REQUIRED</prop>
        </props>
    </property>
</bean>

<!-- Generic DAO - can be used when doing standard CRUD -->
<bean id="dao" class="com.student.dao.hibernate.BaseDAOHibernate">
    <property name="sessionFactory"><ref local="sessionFactory"/></property>
</bean>

<bean id="studentDAOHibernate" class="com.student.dao.hibernate.

```



```

    StudentDAOHibernate">
        <property name="sessionFactory"><ref local="sessionFactory"/></property>
    </bean>

    <!-- Generic manager that can be used to do basic CRUD operations on any objects -->
    <bean id="manager" parent="baseTransactionProxy">
        <property name="target">
            <bean class="com.student.service.impl.BaseManager">
                <property name="DAO"><ref bean="dao"/></property>
            </bean>
        </property>
    </bean>

    <bean id="studentManager" parent="baseTransactionProxy">
        <property name="target">
            <bean class="com.student.service.impl.StudentManagerImpl">
                <property name="studentDAO"><ref bean="studentDAOHibernate"/>
            </property>
        </bean>
    </property>
</bean>

    <bean name="/studentAction"
        class="com.student.webapp.action.StudentManageAction">
        <property name="studentManager"> <ref bean="studentManager"/>
    </property>
</bean>

</beans>

```

程序说明：Spring 框架的配置文件。

程序 6-15: web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
    <context-param>
        <param-name> contextConfigLocation    </param-name>
        <param-value>/WEB-INF/hibernate_context.xml</param-value>
    </context-param>
    <filter>
        <filter-name>encodingfilter</filter-name>
        <filter-class>com.student.filter.EncodingFilter</filter-class>
        <init-param>
            <param-name>encoding</param-name>
            <param-value>UTF-8</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>encodingfilter</filter-name>

```

```

        <url-pattern>/*</url-pattern>
    </filter-mapping>

    <listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener
        </listener-class>
    </listener>

    <servlet>
        <servlet-name>action</servlet-name>
        <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
        <init-param>
            <param-name>config</param-name>
            <param-value>/WEB-INF/struts-config.xml</param-value>
        </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>action</servlet-name>
        <url-pattern>*.do</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <taglib>
        <taglib-uri>/WEB-INF/struts-bean.tld</taglib-uri>
        <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
    </taglib>
    <taglib>
        <taglib-uri>/WEB-INF/struts-html.tld</taglib-uri>
        <taglib-location>/WEB-INF/struts-html.tld</taglib-location>
    </taglib>
    <taglib>
        <taglib-uri>/WEB-INF/struts-logic.tld</taglib-uri>
        <taglib-location>/WEB-INF/struts-logic.tld</taglib-location>
    </taglib>

</web-app>

```

程序说明：Web 应用的配置文件。

程序 6-16: StudentManageAction.java

```

package com.student.webapp.action;
import com.student.webapp.form.StudentForm;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.struts.action.*;

```



```

import com.student.service.StudentManager;
import com.student.model.Student;
import java.io.IOException;
import java.util.*;
import javax.servlet.ServletException;

public class StudentManageAction extends BaseAction {
    private StudentManager studentManager;

    public ActionForward execute(
        ActionMapping mapping, ActionForm form,
        HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException, Exception {
        String method=request.getParameter("method");
        if(method.endsWith("delete"))return delete(mapping, form, request, response);
        else if(method.endsWith("list"))return list(mapping, form, request, response);
        else if(method.endsWith("save"))return save(mapping, form, request, response);
        else if(method.endsWith("update"))return update(mapping, form, request, response);
        else if(method.endsWith("search"))return search(mapping, form, request, response);
        else return unspecified(mapping, form, request, response);
    }

    public ActionForward delete(ActionMapping mapping, ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response) throws Exception {

        StudentManager mgr = getStudentManager();
        StudentForm studentForm = (StudentForm) form;
        mgr.removeStudent(studentForm.getStudentid());
        return list(mapping, form, request, response);
    }

    public ActionForward list(ActionMapping mapping, ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response
        ) throws Exception {

        StudentManager mgr = null;
        List lst = null;
        try {
            mgr = (StudentManager)getStudentManager();
            lst = mgr.getStudents(new Student());
        }
        catch (Exception es) {
            System.out.println(es.toString());
        }
        request.setAttribute("students", lst);
        return mapping.findForward("list");
    }

    public ActionForward save(ActionMapping mapping, ActionForm form,

```

```
        HttpServletRequest request,
        HttpServletResponse response
        ) throws Exception {
    if (isCancelled(request)) {
        return list(mapping, form, request, response);
    }
    StudentManager mgr = getStudentManager();
    StudentForm studentForm = (StudentForm) form;
    Student student = new Student();
    copyProperties(student, studentForm);
    try {
        mgr.saveStudent(student);
    }
    catch (Exception es) {
        System.out.println(es.toString());
    }
    return list(mapping, form, request, response);
}

public ActionForward update(ActionMapping mapping, ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response
        ) throws Exception {

    if (isCancelled(request)) {
        return list(mapping, form, request, response);
    }
    StudentManager mgr = getStudentManager();
    StudentForm studentForm = (StudentForm) form;
    request.setAttribute("student", mgr.getStudent(studentForm.getStudentid()));
    return mapping.findForward("update");
}

public ActionForward search(ActionMapping mapping, ActionForm form,
        HttpServletRequest request,
        HttpServletResponse response
        ) throws Exception {

    String no = request.getParameter("studentid");
    if (no == null || no.length() < 1) {
        return list(mapping, form, request, response);
    }
    ArrayList list = new ArrayList(1);
    StudentManager mgr = getStudentManager();
    try {
        list.add(mgr.getStudent(Integer.parseInt(no)));
    }
    catch (Exception es) {
        System.out.println(es.toString());
    }
}
```



```

    }
    request.setAttribute("students", list);
    return mapping.findForward("list");
}

public ActionForward unspecified(ActionMapping mapping, ActionForm form,
    HttpServletRequest request,
    HttpServletResponse response
    ) throws
    Exception {
    return list(mapping, form, request, response);
}

public StudentManager getStudentManager() {
    return studentManager;
}

public void setStudentManager(StudentManager studentManager) {
    this.studentManager = studentManager;
}
}

```

程序说明：用来实现业务逻辑的主要组件。

说明：为节省篇幅，完整的源代码请参考随书光盘。

运行结果

程序发布成功后，在浏览器地址栏输入“http://localhost:8080/Student/”，将得到如图 6-18 所示的运行结果。



图 6-18 显示学生信息列表

可以看到显示当前学生信息列表。可以单击【添加新生】链接添加新的学生信息，如图 6-19 所示。在【编号】文本框中输入要查询的学生编号来查找指定的学生。单击【编辑】链接，可对相应的学生信息进行编辑，如图 6-20 所示。



图 6-19 添加学生信息



图 6-20 编辑学生信息

讨论与思考

选择框架的目的在于确保应用系统的健壮性，提高开发效率，因此在框架选择时，要根据具体情况灵活运用，不能盲目在项目实践中生搬硬套，堆砌框架。目前适用 Java 企业应用的系统框架可谓百花齐放，各种框架都有长短，选择应用系统框架时不可盲目地追求流行，首先需要明确所要实现的应用系统的系统处理能力需求，然后在熟悉比较各种框架细节的基础上从设计及开发效率方面进行考虑。

知识点索引

架构：Spring；Struts；Hibernate。

本章小结

EJB 的体系结构是 Java EE 的基础和核心，基于 EJB 的框架一度成为人们开发 Java 企业应用的首选。

随着 Java 开源项目阵营的发展壮大,一些基于 POJO 的开源框架被越来越广泛地引入到 Java 企业应用的开发中来。根据复杂程度人们习惯把基于 EJB 的框架称为重量级框架,把基于 POJO 的开源框架称为轻量级框架。

从应用逻辑上,Java 企业应用框架一般被划分为三个层次:表现层、业务逻辑层和持久层。

下面分别对上述三个层次上的架构技术进行分析比较。

1. 表现层

EJB 只用来实现业务层和持久层。常见的表现层开源框架有 Struts、JSF 和 Tapestry。Struts 框架由于出现时间早,所以使用相对广泛,它的社区非常活跃,很容易找到很多现成的开源功能标签以供使用,以及示例程序可供参考。但是它的组件在页面中显示的粗粒度及框架类的限制在很多情况下会表现得过于死板,给表示层的开发带来一些额外的代码开销。JSF 在很大程度上类似 Struts,它是由 Sun 推荐的一种新标准,只是 JSF 的组件概念没有像 Struts 那样必须继承 ActionForm 的限制,JSF 在事件粒度上要比 Struts 细腻。JSF 的另外一个优势就是其身后有 Sun 公司和其他一些大公司的支持。Tapestry 是一个完全组件的框架,Tapestry 的组件可以被套嵌并包裹其他组件,因此可以组合形成一个更大的组件或逻辑页面。组件的行为模式为 Web 页面编程提供了很大的方便,事件处理也方便很多。所以,如果做一个对页面要求灵活度相当高的系统就可以考虑选用 Tapestry。上述三个框架的特性对比如表 6-4 所示。

表 6-4 几种表现层框架的对比

框 架	Struts	Tapestry 3.0	JSF
实现模式	标签库+组件,组件必须继承 ActionForm	完全组件,分显式调用和隐式调用,组件必须继承 BaseComponent	标签库+组件,普通 POJO 无需继承
显示粒度	View 页面只能显示与表单对应的 ActionForm,配置中,Action 与 ActionForm 与页面一般只能是 1:1:1 关系	可将组件嵌入页面任何一行,对使用组件数量无限制	同 Tapestry
面跳转	使用标记 html:link 来实现,其中标记 html:link 的属性 URL 需要与配置文件 struts_config.xml 中的 path 元素名称相对应,而 path 元素被映射到指定的 Action 组件。这样,利用与配置文件 struts_config.xml 作为中介,应用实现了页面跳转 URL 与 Action 组件的松散耦合	页面跳转 URL 名称是目标组件的名称,不涉及 URL 映射等操作,方便稳固	类似 Struts,也需要配置文件作为中介,跳转 URL 地址映射到文件配置信息,配置信息又与组件实现绑定。它同样也实现了页面跳转与目标组件松散耦合
事件触发	通过表单提交 submit 激活,不能细化到表单里字段	能够给予表单每个字段赋一个事件,事件组件必须实现 PageListener 接口	同 Tapestry,事件组件必须实现 ActionListener

2. 业务逻辑层

EJB 2.1 框架有些过于复杂了,有如下缺点:

- ✎ EJB 模型需要建立许多组件接口和实现许多不必要的回调方法;
- ✎ EJB 的部署描述复杂而容易出错;
- ✎ 开发人员不能脱离 EJB 容器测试。

对于以上缺点 JCP 制订的 EJB 3.0 标准框架做了相应的改进,该框架为所有主要的 Java EE 厂商支持。

EJB 3.0 框架与应用服务器高度整合,服务整合代码也包装在一个标准接口后面。EJB 框架一方面有成熟的 EJB 容器支持,基于 EJB 框架的企业应用性能优良;另一方面 EJB 容器设计因为考虑了多方面的功能,所以在其内核上总是会显得臃肿,这也是一种重量表现。不需要的东西存在肯定会影响效率,EJB 不能根据项目需求对 EJB 整体包括 EJB 容器进行可配置式的切割。

Spring 框架处于应用服务器和服务库的上方,服务整合的代码属于框架,并暴露于应用开发者。它与应用服务器整合的能力相对 EJB 3.0 要弱。但是 Spring 框架模块的可分离配置体现了它优于 EJB 3.0 的灵活性。常见业务逻辑层框架的特性对比如表 6-5 所示。

表 6-5 常见业务逻辑层框架的特性对比

框 架	EJB 2.0	EJB 3.0	Spring 1.x
灵活性(松耦合)	不支持应用系统 POJO	支持应用系统 POJO	支持应用系统 POJO, 框架本身可分离配置
功能完整性	支持异步 JMS 分布式事务	支持异步 JMS 分布式事务	较为全面, 有自己的表现层和持久层模板, 可支持异步
IOC/AOP 支持	不支持	支持 IOC, 部分支持 AOP	基于 JavaBeans 类的细粒度支持 AOP
性能	一般, 且大数据量业务处理则性能急剧恶化	一般, 且大数据量业务处理则性能急剧恶化	一般, 应用程序可配置 cache/Pool 以提高性能
可伸缩性	支持多台服务器分布式计算	支持多台服务器分布式计算	不支持
开发效率	低	低	一般
系统规模	大型系统	中大型系统	适应各种系统

3. 持久层

容器管理持久性 (CMP) 是对 EJB 中 Entity Bean 进行持久性管理的方式。EJB 2.1 持久性模型过于复杂并且存在缺陷。EJB 3.0 持久层针对 EJB 2.1 的缺陷做了相应改进, 采用与 Hibernate 类似的机制。

开源的持久层框架主要有 Hibernate 和 iBATIS。Hibernate 相对而言具有如下优势:

- ✎ Hibernate 使用 Java 反射机制而不是字节码增强程序来实现透明性;
- ✎ Hibernate 的使用简单;
- ✎ 映射的灵活性很出色, 它支持各种关系数据库, 从一对一到多对多的各种复杂关系。但 Hibernate 也有一些缺点, 它限制所使用的对象模型, 如一个持久性类不能映射到多个表。

使用 iBATIS 提供的 O/R 映射机制, 对业务逻辑实现人员而言, 面对的是纯粹的 Java 对象, 这一层与通过 Hibernate 实现 O/R 映射而言基本一致, 而对于具体的数据操作, Hibernate 会自动生成 SQL 语句, 而 iBATIS 则要求开发者编写具体的 SQL 语句。相对 Hibernate 等“全自动”O/R 映射机制而言, iBATIS 以 SQL 开发的工作量和数据库移植性上的让步, 为系统设计提供了更大的自由空间。作为“全自动”O/R 映射实现的一种有益补充, iBATIS 的出现显得别具意义。

最后讨论一下在项目实践中选择框架应该注意的几个问题。

设计和性能是实际框架选择的两个基本点, 善于平衡才是框架选择的主要宗旨。轻量级框架和重量级框架解决问题的侧重点是不同的, 因此在选择框架时要根据项目特点有所侧重。

轻量级框架侧重于减小开发的复杂度, 相应的它的处理能力就有所减弱 (如事务功能弱、不具备分布式处理能力), 比较适用于开发中小型企业应用。采用轻量级框架一方面是因为尽可能地采用基于 POJO 的方法进行开发, 使应用不依赖于任何容器, 这可以提高开发调试效率; 另一方面, 轻量级框架多数是开源项目, 开源社区提供了良好的设计和许多快速构建工具及大量现成可供参考的开源代码, 这有利于项目的快速开发。

而作为重量级框架, EJB 框架则强调高可伸缩性, 适合于开发大型企业应用。在 EJB 体系结构中, 一切与基础结构服务相关的问题和底层分配问题都由应用程序容器或服务器来处理, 且 EJB 容器通过减少数据库访问次数及分布式处理等方式提供了专门的系统性能解决方案, 能够充分解决系统性能问题。

轻量级框架的产生并非是对重量级框架的否定, 在某种程度上可以说二者是互补的。轻量级框架在努力开发功能更强大、更完备的企业应用; 而新的 EJB 规范 EJB 3.0 则在努力简化 Java EE 的使用, 以使得 EJB 不仅仅是擅长处理大型企业系统, 也利于开发中小型系统, 这也是 EJB 轻量化的一种努力。对于大型企业应用及将来可能涉及能力扩展的中小型企业应用, 结合使用轻量级框架和重量级框架也不失为一种较好的解决方案。

附录 A 开发环境的搭建

本书所有示例的开发环境采用如下的软件组合: Windows XP SP2+JDK6.0+ Netbeans 5.5.1+ MySQL 5.0

1. 安装 JDK 6.0

JDK 是用于构建发布在 Java 平台上的组件和应用程序的开发环境。它是一切 Java 应用程序的基础,所有的 Java 应用程序都是构建在 JDK 之上。本书使用的 JDK 版本是 6.0 update 2。可以到 Sun 的网站(<http://www.sun.com>)上下载 JDK 最新版本。

说明: Netbeans IDE5.5.1 至少需要 JDK 5.0 以上的版本。

双击安装程序文件 jdk-6u2-windows-i586-p.exe 开始安装 JDK, 如图 A-1 所示,

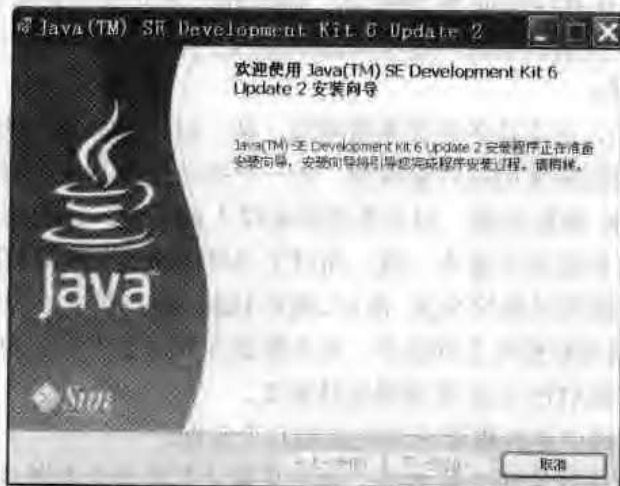


图 A-1 开始安装 JDK

在随后出现的安装选项界面中, 可以设置 JDK 安装路径等信息。默认所有安装选项, 如图 A-2 所示, 单击【下一步】按钮, 开始安装 JDK。

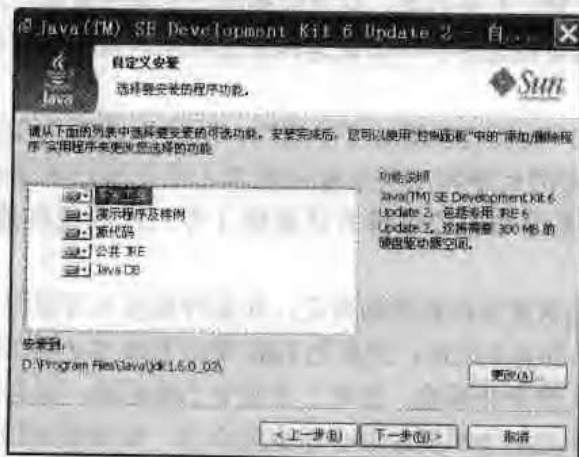


图 A-2 JDK 安装选项设置

在安装过程中，将弹出 Java 运行环境（JRE）选项设置窗口，如图 A-3 所示，默认所有安装选项，单击【下一步】按钮，继续安装过程。

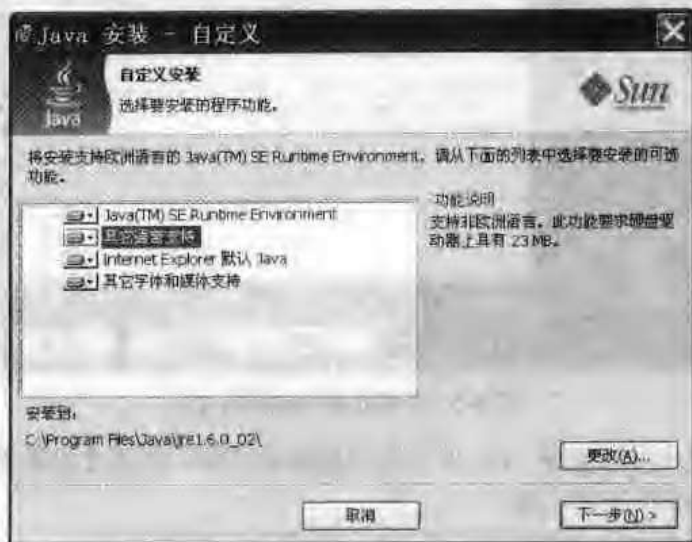


图 A-3 JRE 安装选项设置

最后出现如图 A-4 所示的运行结果，单击【完成】按钮，JDK 安装完毕。



图 A-4 JDK 安装成功

2. 安装 Netbeans 5.5.1

NetBeans IDE 是为软件开发者提供的一个免费、开放源代码的集成开发环境。它可以在多种平台上运行，其中包括 Windows、Linux、Solaris 和 MacOS。NetBeans IDE 易于安装和使用，它为开发者创建专业的跨平台的桌面、企业、Web 和 Mobile 应用程序提供了所需的全部工具。

可以从地址<http://www.netbeans.info/downloads/index.php> 下载最新版本的 Netbeans。本书使用的版本为 5.5.1。它内置的应用服务器为 Sun Java System Application Server PE 9。

双击安装文件，则安装程序自动开始运行，并显示如图 A-5 所示的安装提示界面。

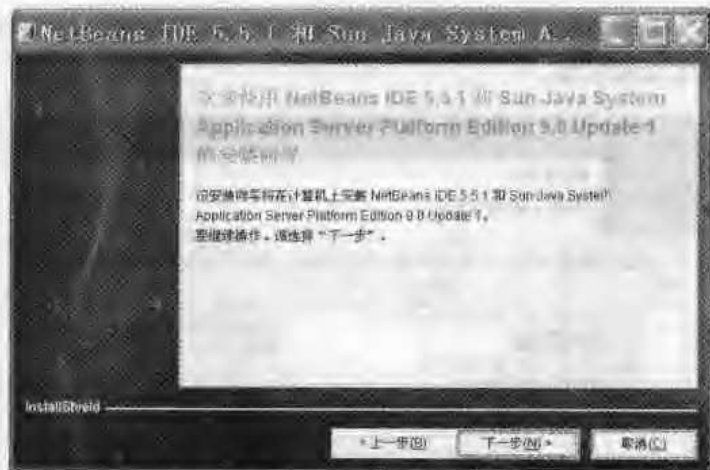


图 A-5 开始安装 NetBeans IDE

单击【下一步】按钮，得到如图 A-6 所示的安装提示界面。选择【我接受许可协议中的条款】，然后单击【下一步】按钮，继续安装过程。

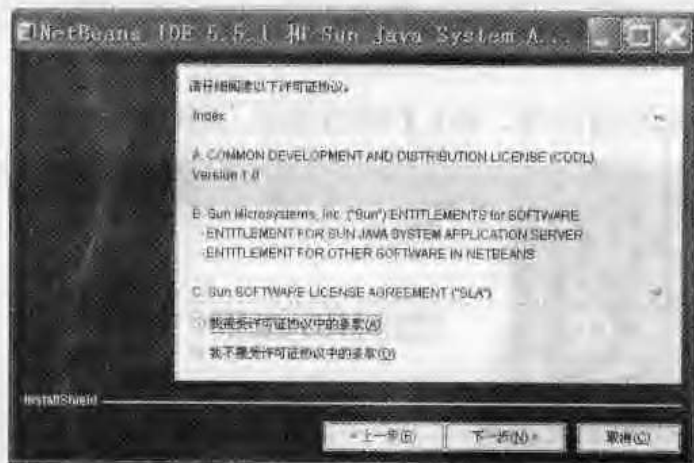


图 A-6 接受使用许可协议

如图 A-7 所示，在弹出的安装提示窗口中，分别单击相应的【浏览】按钮选择 Netbeans 和 Sun Java System Application Server 的安装路径。然后单击【下一步】按钮，继续安装。

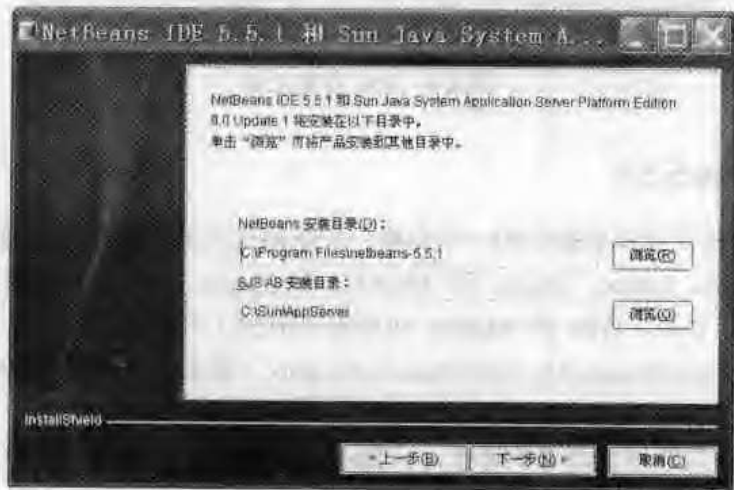


图 A-7 选择安装路径

最后得到如图 A-8 所示的界面，表示 NetBeans IDE 安装成功。

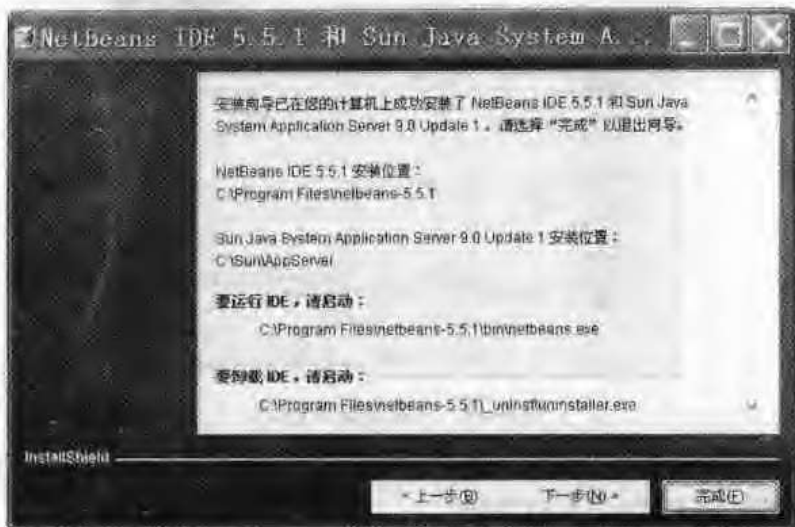


图 A-8 NetBeans IDE 安装成功

3. 安装 MySQL 5.0

MySQL 是由瑞典的 T.c.X 公司开发的一个精巧的 SQL 数据库管理系统。由于它的强大功能、灵活丰富的应用编程接口及精巧的系统结构，受到了广大自由软件爱好者甚至是商业软件用户的青睐，本书所有例程中使用的数据库系统均采用 MySQL 数据库系统。

下面介绍 MySQL 数据库的安装。可以到网站 (<http://www.mysql.com>) 上去下载 MySQL 的安装文件。本书中使用的 MySQL 数据库系统的版本是 5.0.41。

双击安装文件，则安装程序自动开始运行，并显示如图 A-9 所示的安装提示界面。



图 A-9 开始安装 MySQL

单击【Next】按钮开始安装，将得到 MySQL 的安装类型选择界面，如图 A-10 所示。



图 A-10 开始安装 MySQL

单击【Typical】，即选择典型安装模式，单击【Next】按钮继续安装，在随后的安装过程中，默认所有选项，最后得到如图 A-11 所示的运行界面。



图 A-11 MySQL 安装完毕

选择【Configure the MySQL Server now】，单击【Finish】按钮，则 MySQL 安装完毕并启动了 MySQL 服务器配置向导，如图 A-12 所示。



图 A-12 MySQL 服务器实例配置向导启动

在随后的配置过程中，接受所有默认选项，最终完成 MySQL 服务器实例的配置。

Java学习群

72030155

进群可以免费获取视频教程以及每日免费听老师讲课

Java学习群：72030155

好资料应该和你的朋友分享，把这本电子书分享给学Java的朋友，Java群空间：可以联系群主获取更多大型企业内部技术教程。